12-1-2010

emove: Movement Detection and Classification for Tagged Assets on Embedded Platforms

Bhagavathy Krishna

To the University Council:

The Thesis Committee for Bhagavathy Krishna certifies that this is the final approved version of the following electronic thesis: emove: Movement Detection and Classification for Tagged Assets on Embedded Platforms.

Santosh Kumar, Ph.D.
Major Professor

We have read this thesis and recommend
its acceptance:

King-Ip Lin, Ph.D.

Lan Wang, Ph.D.

Accepted for the Graduate Council:

Karen D. Weddle-West, Ph.D.
Vice Provost for Graduate Programs

emove: MOVEMENT DETECTION AND CLASSIFICATION FOR TAGGED ASSETS ON EMBEDDED PLATFORMS

by

Bhagavathy Krishna

A Thesis

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Master of Science

Major: Computer Science

The University of Memphis

December 2010

Acknowledgments

Foremost, I would like to thank my thesis advisor, Professor Dr. Santosh Kumar, who has provided me with many opportunities and challenges over the years. He has consistently given great ideas and criticism, putting great faith in my rising to the task. He has taught me to face the challenges in research through patience and perseverance. He has consistently tested my potential and encouraged me do things which were (almost) impossible.

I would like to thank Dr. Kurt Plarre for helping me in different stages of classifier development and testing. He participated in all brainstorming sessions and provided intellectual contributions to this work. He has always been very critical on the improvement of the work and was always willing to help. It was a great pleasure to work with him. He helped me understand various machine learning concepts and Matlab. He was also patient enough to review this thesis multiple times.

Dr. Prabal Dutta from University of Michigan designed and built the Hermes tag node custom specific for the project. Without the tag we could have not have been successful in developing and testing this system on time and I would like to thank him.

Additional thanks goes to my friends Natasha Velaga, Karthik Pilla, Phani Vajja, Manasa Medikonda, Radhika Kilari, Manohar Potluri and Raghuver Kontham for volunteering in data collection without complaining even at the most odd time of the night.

I sincerely thank all the WiSeMaNet lab members for the support and encouragement through the project.

Finally, I would like to thank my father for his faith, love and sharing his electrical knowledge and my lovable mother whom I promised to spend time with now that this is finished.

Abstract

Krishna, Bhagavathy. M.S. The University of Memphis. December 2008. *emove*: Movement Detection and Classification for Tagged Assets on Embedded Platforms. Major Professor: Dr. Santosh Kumar.

Most of the existing real-time asset tracking systems involves frequent communication of a device placed on the object to be tracked with a fixed infrastructure, which rapidly depletes the battery of the tag, due to the high energy demand of communication. This thesis presents *emove*, a system implemented on a low cost tag node that can be attached to assets. An *emove* tag autonomously detects and classifies movement of assets while being in deep sleep mode for most of its life. It communicates with the infrastructure only when it confirms the movement detected is of interest. Our design emphasizes energy-efficient and compliance free usage. We conducted experiments to evaluate the performance of *emove* in real-life. We found that the accuracy of the classifier for an asset tracking application is 98%; the average time to detect a movement is 22.2 seconds. The average lifetime of the tag is increased from few months to more than a year. We obtained similar results by applying *emove* to a theft detection application.

Contents

List of Tables

List of Figures

1 Introduction

The implications of new technologies and e-commerce on the supply chain in general, and on warehouse management in particular, has been investigated extensively in the last decade [15, 18]. According to a study of the ARC Advisory, the worldwide market for Supply Chain Management (SCM) is expected to grow at a compounded annual growth rate (CAGR) of 7.4% over the next five years. The market is forecast to be 10.3 billion dollars in 2010. One of the processes where automation can lead to large scale savings and better time-to-market is asset tracking.

Often manufacturing processes are adjoined with several large warehouses to store and and manage manufactured items (or inventory) before shipping them to the retailers. Several such industrial complexes are spread out geographically and consist of elaborate in-campus road networks for facilitating material transportation. To ensure proper operation of the supply chain and for other maintenance purposes a large variety of expensive tools and assets are required. Having an asset tracking system across the industrial campus to track various inventory tools and gadgets in and between various warehouses can lead to significant benefits, some of which are highlighted below.

- **Identification of unused, rarely used, and heavily used Assets** - Utilization reports from the asset tracking system will help categorize assets by frequency of their usage (signified by change in location). Unused assets can be eliminated from the active assets inventory to eliminate the cost associated with storing and maintaining them. Low-use assets can be targeted for elimination by the acquisition of dual-purpose equipment. Additional units of heavy-use assets can be acquired and their maintenance prioritized. Since it can be assumed that these assets are critical for the operation of the enterprize, any change in these assets' status (like increase in demand or decrease in availability) can lead to business disruptions. Because of their high use, the maintenance, calibration and up-time of these assets are critical. Identifying high-use assets allows companies to more effectively plan the activities required to keep them operating. It also allows them to purchase more of those tools and pieces of equipment that they use most frequently for projected increases in overall organization activity.

- **Policing timely return** - Sometimes, employees check out a piece of equipment or tool, and keep it longer than necessary. This can cause interruptions in the availability of tools. It can also lead to the acquisition of more tools than reasonably required for the current level of organizational activity. An asset tracking system allows organizations to police timely return of assets for use by other people.

- **Accounting** - Sometimes, organizations charge departments "rent" on capital assets that become part of cost (direct or overhead) of the department using them. Asset tracking systems provide an easy and accurate way of determining the total usage time of each asset by department, to apportion the related costs fairly.

- **Depreciation Life of Assets** - Knowing the current non-depreciated value of your assets can help organizations to make repair and maintenance decisions. Often, repair costs will exceed the remaining value of an asset. Then they will have more information in making capital expenditure decisions.

- **Centralization of Asset Detail Information** - Scheduled Maintenance, Calibration Requirements, Software/Firmware Revision, and other detailed asset information can be pulled up by scanning the asset barcode or keying in the asset number. Inventory managers can run reports on maintenance due, calibration due etc. This ability makes scheduling any type of maintenance easy.

In addition to improving supply chain management, asset tracking can benefit all institutions and businesses who maintain assets that move across buildings or warehouses. For example, asset tracking can help enhance patient care by helping medical staff quickly find required devices, reduce the loss of equipment, size up equipment inventory, improve customer service by making it possible to quickly locate a requested equipment, accelerate reaction time in emergency situations, etc. However, for the large scale adoption of any asset tracking system it needs to be affordable and long lasting to avoid repeated user compliance to change or recharge embedded batteries.

Most existing technological solutions currently available for asset tracking are radio frequency identification (RFID)- and wireless local area network (WLAN)-based systems and a vehicle security system from LoJack [1]. An RFID-based system consists of RFID tags and RFID readers for detecting the tags. Passive RFID tags operate without batteries and hence have a very long lifetime, but very short communication range. Hence, they can only be used in asset tracking on conveyer belts where the readers can be placed in close proximity to the passing tags. Active RFID tags have longer communication range and can be used for asset tracking in large areas and outdoor environments. Wisetrack [30], GAO [31] and Tagsys [32] are examples of widely used RFID-based systems for asset tracking. WLAN based systems such as Horus [29] and MagicMap [14] are an alternative to active RFID-based systems that also regularly communicate with the access points. The LoJack system deploys/leases cellular towers for establishing continuous long-range communication between the tag in the car and the tower. All the above systems communicate with the infrastructure periodically

to provide its location information.

Ideally, an asset tracking system should be affordable, small enough for embedding into assets of various dimensions and have minimal compliance requirement from the owner (such as no need for battery recharging, no need for manual reporting, etc.). Most existing asset tracking systems depend on frequent communication with the tracking infrastructure to detect change in locations (e.g., GPS satellites for GPS based systems, and RFID readers, WLAN access points, or cell towers in others), which rapidly depletes the battery and requires charging every few weeks unless the tags have a continuous source of power such as car battery in LoJack. In all the RFID and WLAN based systems, the tags always remain awake and rapidly depletes the battery of energy within weeks, while the LoJack system lasts about three days if it loses access to the car's battery. LoJack system is primarily built for tracking stolen cars and uses a long range radio which drains battery quickly. Additionally, the weight and dimension of the LoJack system makes it difficult for plugging it into even mid-sized inventory tools rendering it unsuitable for asset tracking purposes.

In this thesis, we present **emove**, a new movement detection and classification system for tagged asset. *emove* initiates the communication of the asset to the infrastructure only when the asset is moved, thus reducing the energy consumption significantly and increasing its overall lifetime. The *emove* system consists of a battery-powered **Tag node** that is to be attached to valuable assets which are to be tracked. Assets like expensive inventory maintenance tools are to be tagged using these tag nodes to reveal their current locations, usage patterns, etc. The hardware prototype of the *emove* tag was built based on the Epic Core [11] wireless sensor mote platform. The design of the tag node emphasizes extremely low power so that it can last 10 years on a coin cell battery, ultra-low cost so that organizations may purchase hundreds of them, and tiny form factor so that it can be embedded into a wide variety of assets.

The problem is to develop a low cost tiny tag that can be attached to static assets (eg., televisions, printers, gaming systems, projectors) to detect and identify movement using ultra-low energy. Developing an efficient and reliable algorithm to detect movement with limited resources and energy is a challenge. To address this challenge, we use the following approach.

- The tag detects all movements, yet remains in deep sleep mode most of the time remaining insensitive to natural vibrations.
- The movement detection and classification is done in a hierarchical manner. More complex methods are involved only after simpler methods confirm that the movement detected may be of interest.

- Communication is established only when the event of interest (movement) occurs as "radio is expensive" in terms of energy.

The tag node detects movement autonomously using a hierarchical wake-up system of passive and active vibration sensors. To maximize the time between recharging to years, we use vibration dosimeter so that the tag nodes can spend most of their life (i.e., when not being moved) in an ultra-low power state, drawing only nanoamperes of current. When experiencing movement or jerks, the tag samples a low-power accelerometer and uses machine learning based algorithms to determine if the tag is indeed being moved and also classifies the type of movement. We applied *emove* to a system which tracks assets called AssetTrack. The AssetTrack system consists of two major components, a low cost tag node that is to be attached to assets and a scalable industrial-campus wide infrastructure of anchor nodes to track the movement of tagged assets (via radio communication between anchors and tags). In AssetTrack the radio needs to be is duty cycled with different frequency to send messages according to the type of movement during tracking. These messages provides real-time updates on the current location of mobile asset to inventory management unit when moved. Using *emove* in AssetTrack to detect and identify the movement of interest reduces energy consumption by setting the frequency at which the message is sent according to the type of movement (low frequency for slow movement and high frequency for fast movement), as opposed to communicating at the same rate for all movement types.

emove was also applied to a theft detection system called AutoWitness. The main function of the AutoWiness system is to detect theft and track stolen item. In AutoWitness, theft is associated with vehicular movement, i.e., we assume that a stolen object is transported in a vehicle. *emove* aids in identifying vehicular movement from all other movements and activates the tracking mode on detection of vehicular movement. During the tracking phase distance is estimated between successive stops. Here, *emove* is used to detect when a moving vehicle makes a stop.

Evaluation: We evaluate *emove* by assessing the accuracy of detection, time to detection and energy consumption. We conducted experiments to evaluate the performance of *emove* in real-life. We are able to achieve over 98% accuracy while using 2 simple features derived from accelerometer, all while operating on ultra-low power for majority of the time; the average time to detect a movement is 22.2 seconds and the maximum can go upto 35.2 seconds. The average lifetime of the tag is increased from 5 days to more than a year. We obtained similar results by applying *emove* to a theft detection application.

Contributions: In this thesis, we make two main contributions.

- We propose a systematic approach of hierarchical movement detection and classification system has been developed by using simple machine learning algorithms for establishing/confirming movement of interest in energy constrained devices.
- We evaluate the applicability and performance by implementing *emove* in real life asset tracking system called "AssetTrack" to detect and classify movement of interest for establishing adaptive duty cycling of radio. In theft detection system "AutoWitness", *emove* was implemented as a theft detector and a vehicle stop detector.

1.1 Organization

Section 1 begins by explaining the motivation behind this research on the movement classifier for ultra-low power devices and explains the reason of why the current techniques which is used in many systems is not quite suitable for such energy constrained devices. Section 2 describes the general architecture of the *emove* system. Section 3 discusses the hardware design and the components which was added specifically to meet the requirements of low power classification. Section 4 discusses the basic design cycle for the development of the classifier in *emove*. Section 5 describes the development of *emove* and its application in AssetTrack. Section 6 describes the application of *emove* in AutoWitness as theft detector and stop detector. Section 7 concludes by describing the performance and applicability of *emove* and the challenges which remain.

2 General Architecture

Continuous communication through radio is one of the major factors for depletion of energy. The main idea behind *emove* is to avoid communication when not moved. Figure 1 shows the general architecture of the *emove* system. The tags are deployed in all static assets such as television, washer, piano that needs to be tracked. The tag will be in low power mode until it experiences significant movement. If a movement is confirmed by the tag then it samples the accelerometer for acceleration data. This data serves as an input to the movement classifier. The movement classifier verifies whether the movement is of interest and performs a desired action like sending a message to the infrastructure or triggering a tracking module, according to the application it is used. If the tag is woken up by movements caused by natural vibrations, jerks, quick displacement of the object or any sudden vibrations, the classifier confirms the activity is not an event of interest and the tag again goes to its deep sleep state.

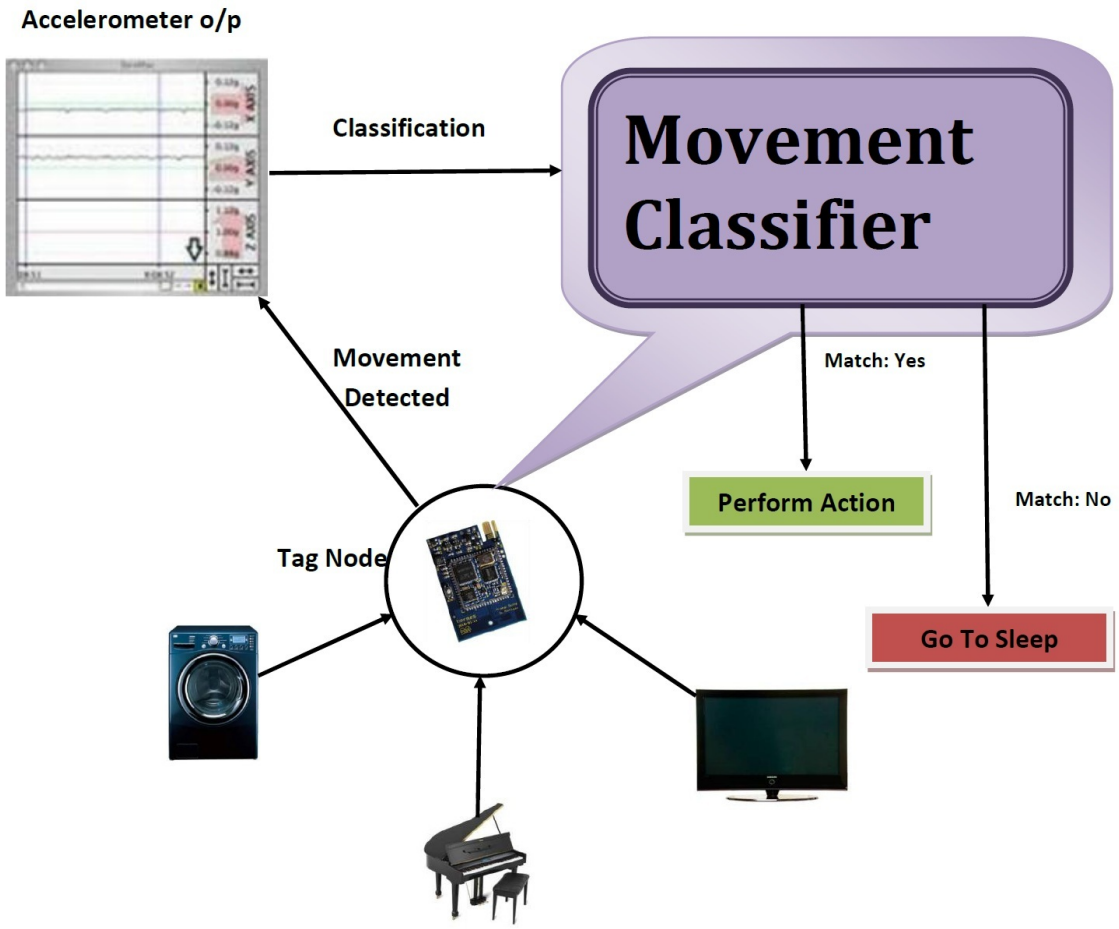
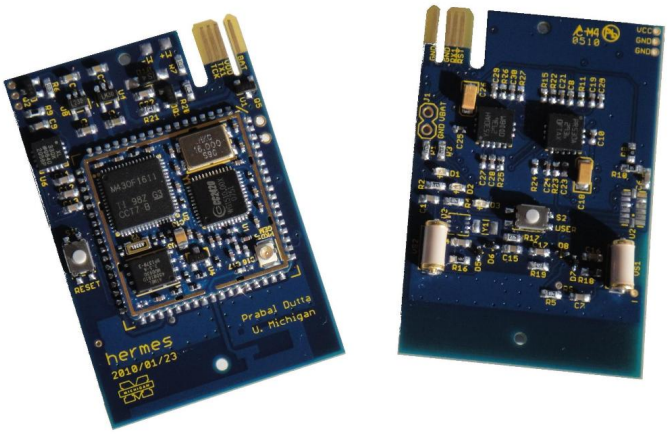


Figure 1: General Architecture of emove



(a) Hermes Tag

Figure 2: Prototype tag hardware. (a) The hermes tag integrates an Epic Core, dual vibration switches, an accelerometer, and a rechargeable Li+ battery in a 51 mm x 34 mm x 10 mm footprint.

3 Hardware

We designed the hardware to address the following challenges: Small size, low energy. Tags must be small and unobtrusive so that they can be deployed in everyday objects like computers, televisions, and stereo equipments. This basic requirements is addressed with the tag design that integrates an epic core, shown in Figure 2(a) with vibration dosimeter and accelerometer. Once deployed, tags must operate unattended for many years without maintenance.

3.1 System Overview

The Hermes hardware was designed and built by Dr. Prabal Dutta from University of Michigan. Some parts of the hermes description was specifically adapted from his writing [13].

The existing design of Hermes, shown in Figure 2(a), has a 51 mm x 34 mm x 10 mm footprint (without the GPRS modem), the entire tag draws over 10 μA in sleep mode.

Figure 2(a) shows hermes a new tag node based on the Epic Core [11] design. Hermes represents a near approximation of our long-term tag node that is available today. Table 1 shows the main components that are used in hermes. Hermes includes a vibration wakeup circuit and 3D accelerometer, which are used as a low cost inertial sensors for movement detection and classification. The layout of hermes circuit with epic core and the inertial sensors can be found in the Figure 7. In the following, we discuss how the principal system components, including the battery, sensors, and

Table 1: The components used in hermes and its model number.

Component	Model
Microcontroller	MSP430F1611
Radio	CC2420RTCR
Flash	AT45DB161D-MU-2.5
Accelerometer	ADXL 330
Vibration Dosimeter	SQ-SEN-200

processor/radio hardware are affected by the constraints of our application.

3.2 Choice of Intertial Sensors

Today, many applications use accelerometers for detecting the onset of motion, however commercially-available accelerometers draw too much power for our needs, even when duty-cycled, and they provide a higher resolution than needed. In addition, the accelerometer output would need to be integrated over time, either in hardware or in software, to detect prolonged motion. These drawbacks discouraged us to use accelerometer as a motion detection sensor in our design, but we do use an accelerometer for confirming movement and detect the event of interest by classification. In future, we plan to explore ways to accomplish these functions without an accelerometer. Another potential motion detection sensor is the piezo-electric vibratab, like the one used in the CargoNet node [19]. Vibratabs and their processing circuits can operate on nano-power budgets, however these sensors and their active integrator circuits add size and cost, which are at a premium in the design of the tag.

Accelerometer: ADXL330

The ADXL330 is a small, thin, low power, complete 3-axis accelerometer with signal conditioned voltage outputs, all on a single monolithic IC. The product measures acceleration with a minimum full-scale range of 3 g. It can measure the static acceleration of gravity in tilt-sensing applications as well as dynamic acceleration resulting from motion, shock or vibration. The user selects the bandwidth of the accelerometer using the CX, CY, and CZ capacitors at the XOUT, YOUT, and ZOUT pins. Bandwidths can be selected to suit the application with a range of 0.5 Hz to 1600 Hz for X and Y axes and a range of 0.5 Hz to 550 Hz for the Z axis.

The key features of ADXL 330 is 3-axis sensing, small and low-profile package (4 mm * 4 mm *

1.45 mm) as shown in Figure 3, low power 180 μA at VS = 1.8 V (typical), single-supply operation 1.8 V to 3.6 V, 10,000 g shock survival, excellent temperature stability, band width adjustment with a single capacitor per axis [2]. Figure 4 shows the functional block diagram of ADXL330.

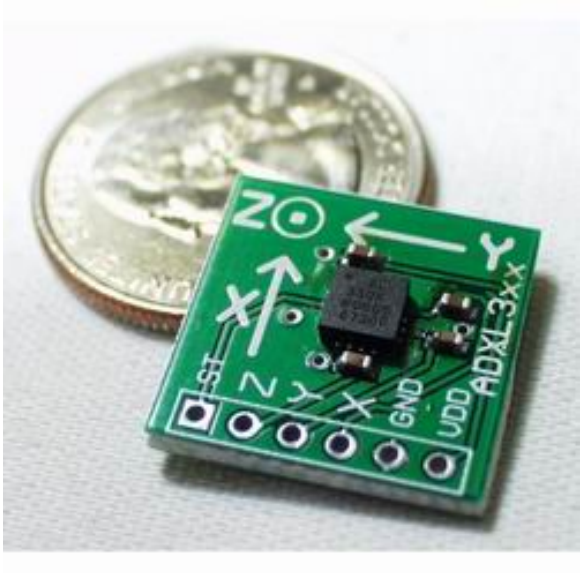


Figure 3: The Accelerometer (ADXL330) is so small and is almost the size of a quarter

Vibration Dosimeter: SQ-SEN-200

The SQ-SEN-200 series sensor as shown in Figure 5 acts like a normal closed switch which chatters open and closed as its tilted or vibrated [24]. Unlike other movement detector sensors the SQ-SEN-200 is truly an omnidirectional movement sensor, which works regardless of its orientation. The key features are ultra low power of 2μA of current, miniature size (3.3 mm x 6.9 mm), high sensitivity, long life, quiet, simple interface which does not require signal conditioning.

The switch in SQ-SEN-200 is not guaranteed to be closed even when the sensing mechanism is under complete rest [24]. Depending on the sensors orientation, the sensor will be closed under rest 75%-95% of the time. So its intuitive to observe the edge transitions like high-to-low or low-to-high transitions rather than an open or closed state of the switch for our detection.

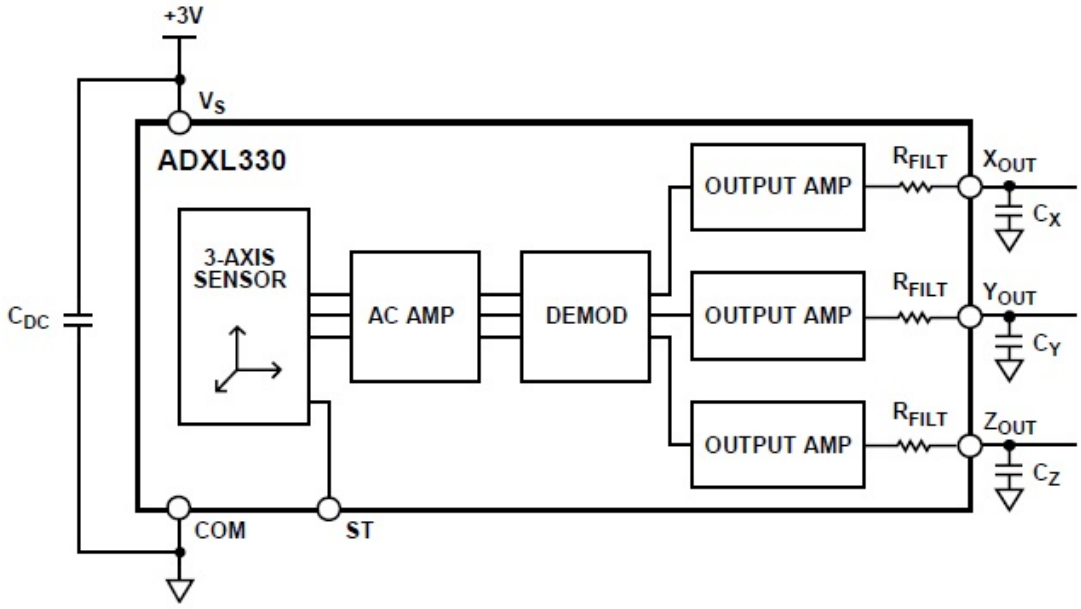


Figure 4: The functional block diagram of ADXL 330

3.3 Core Components

The core components of emove tag-mote have been adapted from other successful platforms and represent a basic system required to perform a low power movement detection and classification.

Epic Core: Epic is an open mote platform which aids application-driven design [11]. Usually sensornet platforms are tightly coupled to their applications. Using general purpose platforms to address application specific needs were almost impossible. It prevented reusability of modules and forces platform designers to re-implement the functionality often. As epic platform overcomes these limitations it became an obvious design choice for tag-nodes. Epic platforms are organized around a general-purpose core module as well as optional peripheral modules. The core module, shown in Figure 6, is essentially the heart of a tag node without the constraints on how it can be used. The core module integrates a state-of-the-art microcontroller, IEEE 802.15.4 radio and flash memory.

Microcontroller (MSP430): The microcontroller used in epic core is MSP430F1611. It offers more memory, better performance and many new features compared to its competitors. The major reasons behind its usage in epic core are being low active current, wide operating range, a 16-bit sleeper, fast wakeup from sleep, large amount of RAM and three direct memory access (DMA)

Figure 5: The vibration dosimeter (SQ-SEN-200) is placed on a quarter to show its miniature size

channels that can operate when the CPU sleeps. Technical specifications of the microcontroller can be found in its datasheet [27].

Radio (CC2420): The CC2420 is a single-chip 2.4 GHz, IEEE 802.15.4 compliant RF transceiver designed for low-power and low-voltage wireless applications. It operates on supply voltage between 2.1 to 3.6V. The effective data rate of CC2420 is of 250 kbps. It complies with worldwide regulations and is region free. The CC2420 provides extensive hardware support for packet handling, data buffering, burst transmissions, data encryption, data authentication, clear channel assessment, link quality indication and packet timing information. These features reduce the load on the host controller and allow CC2420 to interface low-cost microcontrollers. The configuration interface and transmit / receive FIFOs of CC2420 are accessed via an SPI interface. CC2420 can be used together with a microcontroller and a few external passive components. Additional details of CC2420 can be found in its datasheet [7].

Battery: The key battery selection criteria are self-discharge rate (which affects shelf-life), energy density (which affects size), and cost (which affects viability). The common lithium mangenese dioxide ($LiMnO_2$) primary cell provides a good mix of features well-suited to this application. These batteries exhibit a shelf-life of over 10 years at room temperature and are often used as a permanent component for the entire lifetime of a system. Their bulk volumetric energy density is approximately 600 mWh/cm^3, although for some small batteries like photo/coin cells, the effective volumetric energy density can be lower due to packaging overhead. Commonly-available lithium coin cells in the CR family, like the CR2032, are widely-used in consumer products, making them relatively inexpensive.

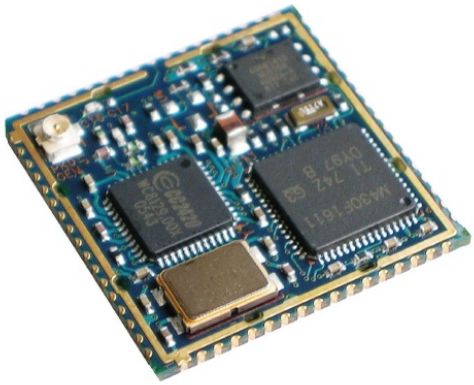


Figure 6: The Epic core module: a wireless sensor tag-node core that integrates a microcontroller, radio, and flash memory.

Although alkaline primary cells also have low self-discharge rates, their volumetric energy density is half of the lithium primary cells, which increases size and their terminal voltage drop makes voltage regulation more important. Common secondary (rechargeable) cells like NiCad, NiMH, Li+ and LiPoly chemistries have higher self-discharge rates, less than half the energy density, and a higher cost per watt-hour than lithium, making them ill-suited to this application.

The Energizer CR2032 has a 10+ year shelf-life (losing only 15-20% of its capacity at room temperature), provides an energy density of 653 mWh/cm^3 (supplying over 200 mAh in a 1 cm^3 package), and available for less than $1 through retail channels (and substantially less in bulk) [12]. These figures translate to approximately 2.5 μA-decade/cm^3 charge density which implies that the average current draw must be less than 2μA to achieve a 10-year lifetime. The energy consumption of various components in hermes is found in Table 2.

4 Design Cycle of a Classifier

Pattern classification has a traditional approach which is followed widely [9]. The design of a pattern classifier can be divided into various stages. Figure 8 shows the widely followed classification process. We observed that it is important to collect the right type of data. Data must be collected for both training and testing the system. The characteristics of the data impact the choice of discriminating features and the choice of models. The training process determine the system parameters. The results of the evaluation may call for repetition of various steps in this process to obtain satisfactory

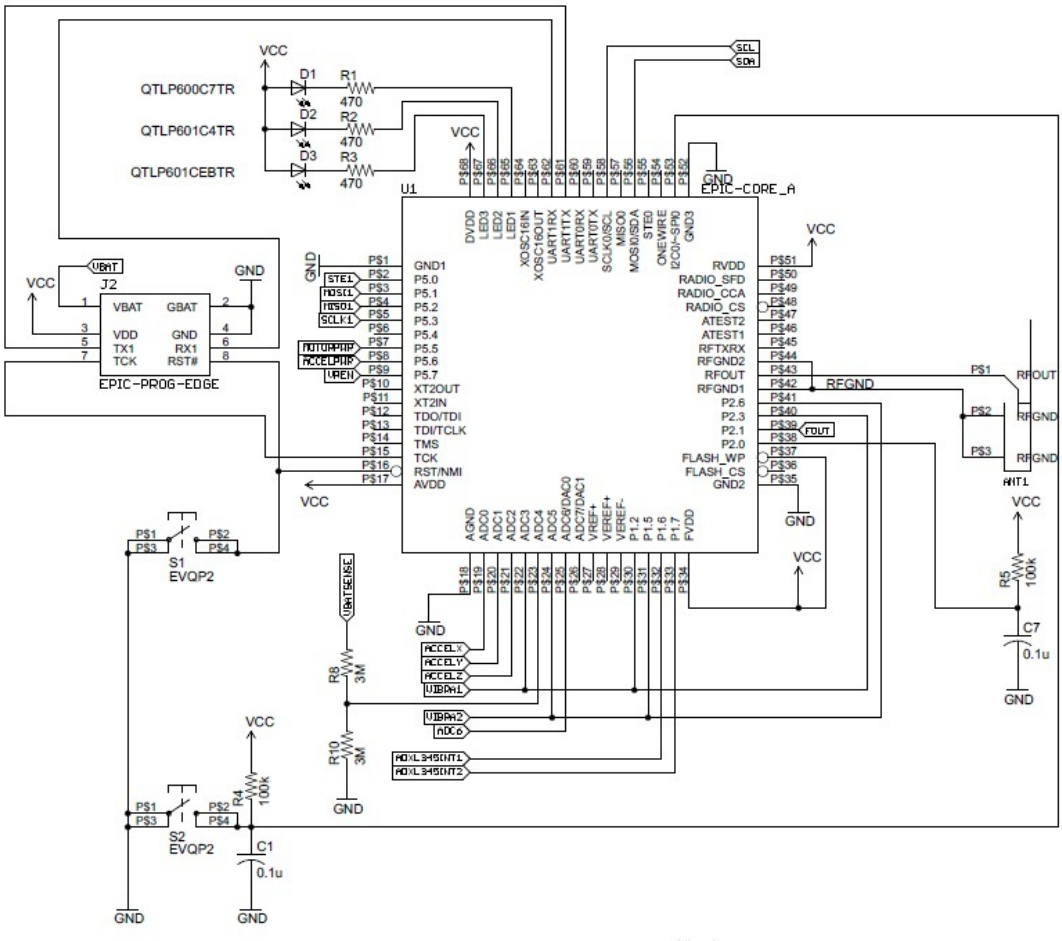


Figure 7: Layout of hermes circuit based on epic core [10]

results. In our case, the traditional classification process did not work as expected. We had to try various pre-processing techniques after data collection step and retrain the classifier for evaluation. We finally also added the post-processing step to the classifier which improved the performance of the classifier significantly.

4.1 Data collection

Data collection is a key step in building a reliable classifier. The performance and the accuracy of the classifier largely depends on the quality of data. To develop a reliable classifier, data was first collected from different activities. The data should be trimmed and tailored for any inconsistences. Usually the first 2 minutes and the last 2 minutes of the collected data is removed. This is done to

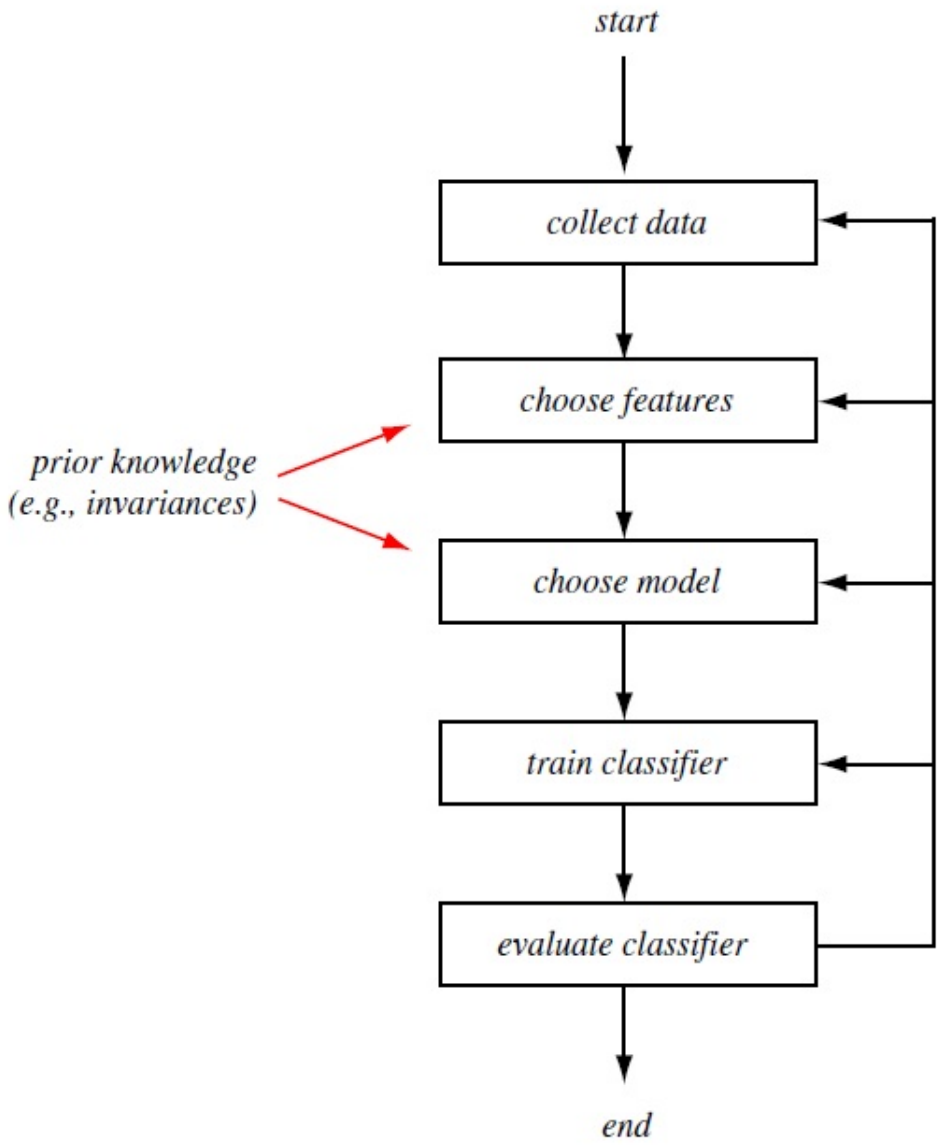


Figure 8: Pattern classification stages taken from [9]. Copyright 2001 by John Wiley and Sons, Inc.

Table 2: The energy consumption of various components of hermes.

Parameter	**Current Draw (μA)**
Microcontroller sleep	2.6
Microcontroller active	500
Radio sleep	1
Radio receive	19700
Radio transmit	17400
Flash sleep	5.4
Flash read	7000
Flash write/erase	12000
Accelerometer	320
ADC	800
Vibration Dosimeter	2

remove any inconsistences or transitions experienced by the tag during data collection. We collected data and is categorized into different classes of interest. Figure 9 shows how the activities are classified into different classes from class 1 to class n. The various activities are grouped together in one of these classes. Now lets look into the basic setup for data collection.

Data collection setup: We attached the tag nodes to the assets of interest. The ADC of the inertial sensors like the accelerometers and vibration dosimeter was sampled continuously at a frequency of 200HZ. The collected data was then transferred to the base station. For the base station, Crossbow's telosb motes [20] were used as receiver. We used the standard CC2420 radio to send and receive the data, however this method was later discarded as it caused severe packet loss and affected the quality of data when sampling for higher frequencies. In some cases the tags were attached to the assets from which data needs to be collected and the tags stored the sampled data in the flash memory. The data was later retrieved from the flash to a PC, for analysis and training. Once the required data is collected the next step would be the selection of right features for classification.

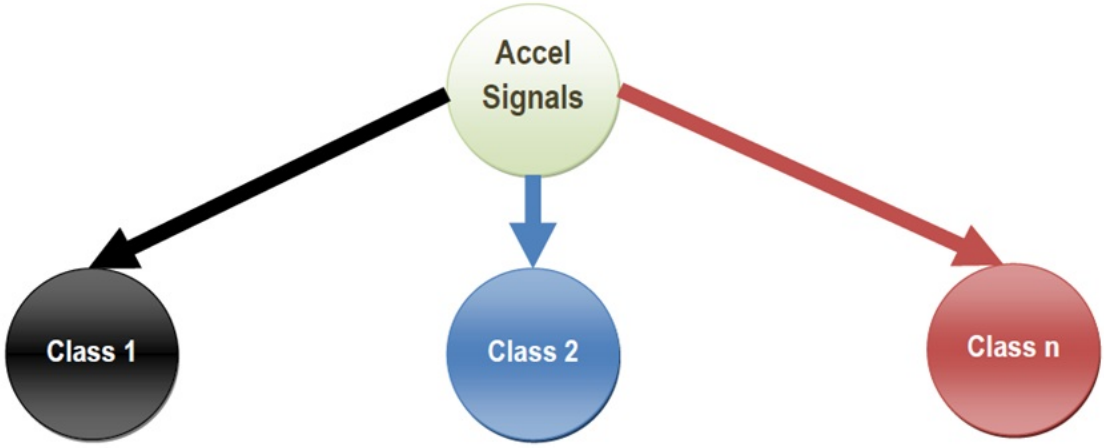


Figure 9: The accelerometer data is divided into different classes of interest from 1 to n

4.2 Feature Selection

We chose many features from various papers, some of them are from [21–23] in which they classify different activities related to movement and they are also used extensively in various applications. From the entire list of features gathered, we selected few features from intuition and other features were selected due to repeated usage in literature. Below is the list of few features which were used to train the classifier. Mean, variance, max minus min, variance normalized, average absolute first difference, variance of first difference zeros removed, energy, low frequency energy, medium frequency energy, high frequency energy, standard deviation, average absolute second difference, average first difference, average absolute first difference normalized, median. We used Weka [28] to access the performance of these features on the collected data set.

4.3 Classifier Selection

Selection of the right type of classifier is critical for the energy constraint tags. The selected classifier should be simple and easily adaptable by these low power tag-motes. We used Weka [28] to analyze the performance of various classifiers on the collected data set. Table 3 shows the performance of various classifiers. From the table we can see that rotation forest and classification via regression perform better but they are too complex to implement on tags. The performance of decision tree (DT) on the other hand is close to the performance of rotation forest and classification via regression. Since our goal is to design a simple classifier that can be implemented on the tag-node, we decided

Table 3: The performance of various classifiers in WEKA.

Classifiers	**Accuracy (%)**
Decision Tree	98.5813
Random Tree	98.4217
Nave Bayes	87.5155
ClassificationViaRegression	98.72
Filtered Classifier	96.1341
Rotation Forest	98.88
Decision Table	95.95

to use DT with the best performing features on the accelerometer data. A DT has an additional advantage over other classifiers for embedded sensor nodes. The sensors can be woken up and features can be computed "as needed", then nodes of the tree can be evaluated with the computed features [5].

4.4 Pre and Post Processing

Following the traditional approach of classifier development as shown in the Figure 8 did not lead to the significant improvement of the classifier. The raw signal was quiet noisy with natural vibrations, jitters and outliers. We had to add a step of pre-processing to the process of classifier development which improved the quality of the signal leading to improvement of the classifier performance. We used low pass filters like butterworth filter to remove unwanted noise in the accelerometer signals. When we trained the classifier with this data set, the classifier was insensitive to few activities especially the car movement. The car engine generates low frequency vibrations which was filtered by butter worth filter. Another approach of applying median over every 15 samples helped in eliminating outliers and unwanted noise while retaining the nature of the signal. It also improved the performance of the classifier significantly. The accuracy of classification, however increased drastically after adding post processing step. The details of the post processing can be found in Section 5.1.4. This method decreased the number of misclassifications and helped in reducing the false positives.

5 Asset Track

The movement classification method described in Section 4 was used in the classifier development of the AssetTrack system. AssetTrack is the system developed to track assets. It has two major components.

1. Low cost tag node attached to assets.

2. Scalable industrial-campus wide infrastructure of anchor nodes to track the assets.

AssetTrack tag-mote operation: Upon deployment (i.e., embedding in an asset), a tag node is registered in a web-based asset tracking system to create association with the corresponding asset. Figure 5 depicts the transition of a tag node in the various states together with events that cause the transition among states. Transition among the *deep sleep*, *movement detection*, *movement classification* state are described in Section 5.2.

Initially, a tag is in a *deep sleep* state (with just a passive vibration switch active). It wakes up when interrupted by the vibration switch (as a result of significant movement, e.g., jerk, displacement, etc.). Once awake, it samples the accelerometer and uses a movement detection algorithm to determine whether it is being actually moved or not (i.e., *movement detection* state). If not, it returns to the *deep sleep* mode. Otherwise, it samples the accelerometer to determine the type of movement and uses a movement classification algorithm to determine the type of movement (i.e., *movement classification* state). If the movement classified is of type *static* then it returns to deep sleep mode. If the movement classified is of type *slow* or *fast* then it sets the radio duty cycling frequency as α or β respectively. It then searches for an anchor node (i.e., *anchor search* state) by transmitting exploration messages at the frequency set and goes back to the *movement classification* state if it exceeds the quota of exploration messages.

5.1 Classifier Development for Asset Track

As described in Section 5, the tag nodes are in *deep sleep* state when not moving. Upon being interrupted by the vibration dosimeter, the tag node needs to determine if it is being moved (i.e., the *movement classification* state), and if so, needs to find an anchor node to facilitate tracking (i.e., the *anchor search* state). This must be done with minimal use of energy, and in a short distance, if possible.

The movement detection algorithms must address the following questions for a tag: (1) when to start transmitting, (2) how frequently to transmit, and (3) when to stop transmitting?

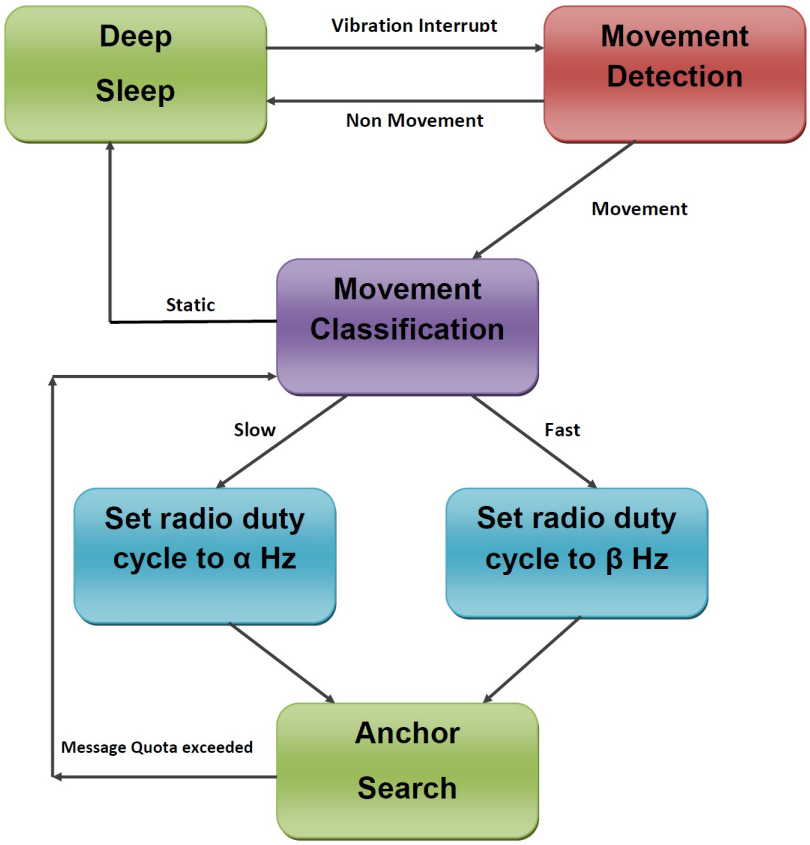


Figure 10: The state transition diagram for a tag node with *emove* in AssetTrack

The tags should not communicate in the static phase. It should remain in its low power state until it has been moved considerably. The frequency of data transmission depends on nature of movement experienced by the tag. Assume a person walking with an tagged asset and the anchor node to which the tag must communicate is at a distance of 10 miles from where he started. A person might be walking at an average speed of 5mph and it might take two hours for him to reach his first anchor node. If the same asset is placed in the car and assume its moving in an average speed of 50mph then it would only take 12 minutes to reach the anchor node. Hence the frequency of the message sent when the object is in an fast moving vehicle should be more as it reaches the anchor more quicker than the slow movements like walking or running. To solve this problem we categorized the types of movement into three classes. (1) Static: In which the asset remains in a state of complete rest and occasionally might experience minor vibrations due to natural factors; (2) Slow: Any movement the asset experiences in manual transportation like walking, running, climbing the stairs, jumping, asset moved on a trolley or on a chair with wheels are considered slow; (3) Fast: All vehicular movement is considered as fast movement as on an average they move with the speed which usually cannot be achieved by any human activity. Figure 11 shows how the frequency of messages should increase of decrease according to the type of its movement. Coming up with an algorithm and frequency at which message needs to be sent does not lie in the scope of this thesis. The scope is restricted to building the energy efficient wake up and classification scheme which triggers the message sending algorithm.

The three classes of interest in this scenario to build a classifier would be slow, static and fast. Figure 12 shows the three classes of interest.

5.1.1 Data Collection for AssetTrack

Data collection is a key step for building a classifier. As mentioned in the Section 4.1 the first step in building the classifier would be to find the classes of interest for AssetTrack. In AssetTrack the classes of interest was the speed in which the object moves which are *"static", "slow", "fast"*. Figure 12 shows how the various activities are categorized in to different classes of interest. Next step would be to place the tags in different type of assets and perform various activities for collecting data. In the Figure 13(a) we can see the Hermes tag with a battery that was attached to various assets. Figures 13(b) 13(d) 13(c) shows how the tag was attached to the assets during data collection. Figure 13(e) shows the mote being attached to a persons arm and data was collected from them during activities like running, walking and climbing.

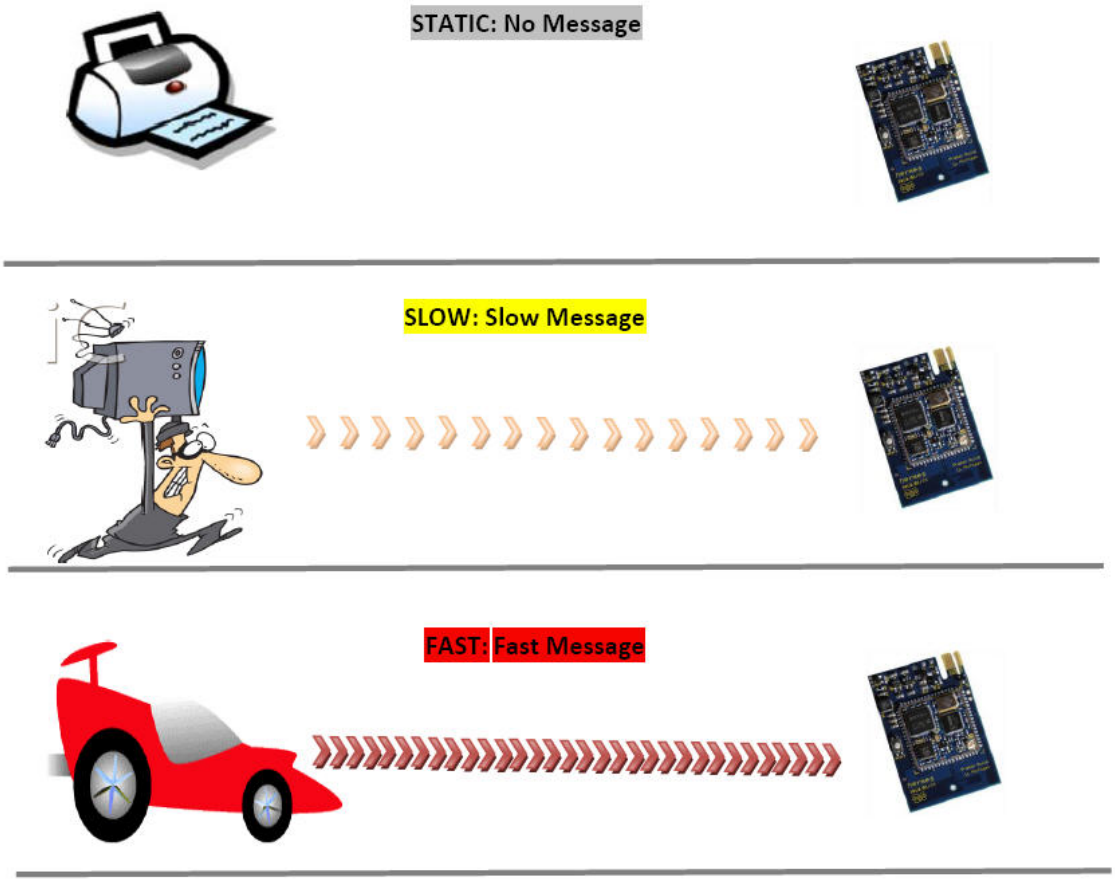


Figure 11: Figure shows the change in frequency of data sent to the anchor mote according to the type of movement

5.1.2 Feature Selection and Classifier Development

The data collected from the sensors are ADC values. The ADC values of all the 3 axis (x, y, z) are converted to acceleration in units of 1g by (a-2048)/245 where a is the ADC value. We then computed the magnitude of acceleration in order to eliminate the dependence of the algorithm on orientation of the tag. We used the magnitude of the acceleration vector for computing the accelerometer features. This is the most common practice when working on random and continuously changing orientation [23]. Figures 14 shows the magnitude of acceleration in units of 'g' for various activities. We sampled the data at a frequency of 200Hz. We used 1.05 seconds of data as a data set to train the classifier. Each data set contained 210 data points. WEKA was used to train the classifier and for selecting the features. Commonly used 10-fold cross validation was used to train and test the

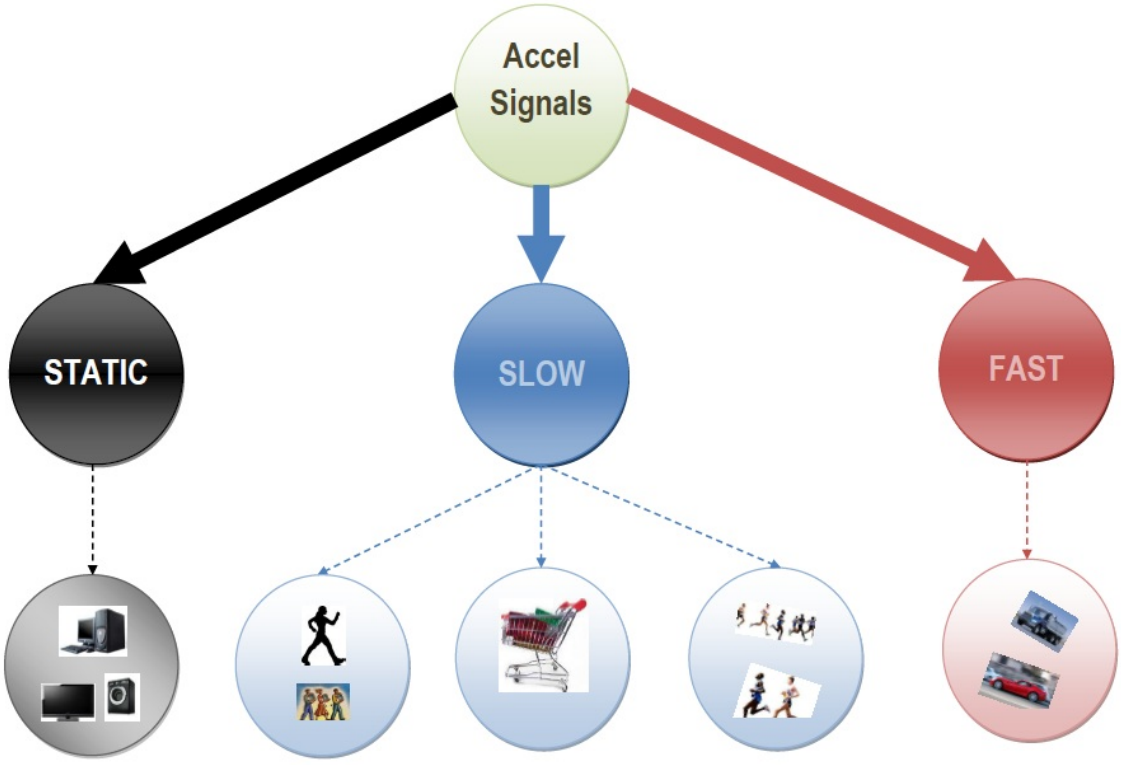
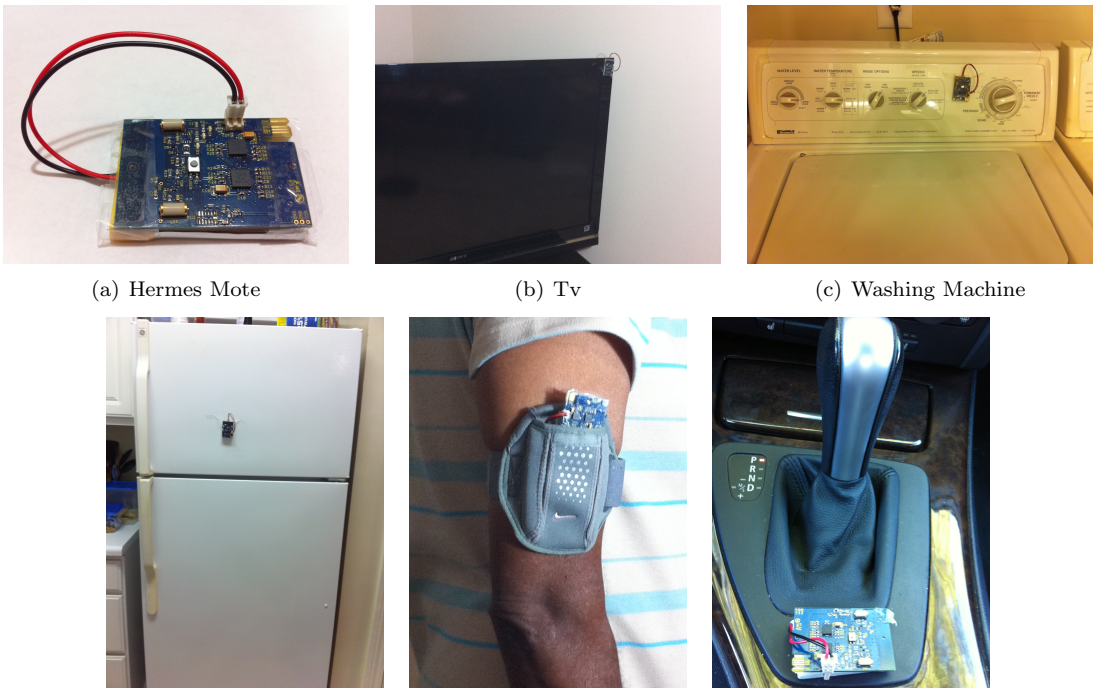


Figure 12: We can see the three classes of interest of AssetTrack and the activities that belong to the corresponding class

DT. Classification is based on the best performing features that are computed from data. We found the most discriminating features were energy (i.e., mean over square of measurements), standard deviation and average absolute second difference. Adding more features did not lead to significant increase in accuracy. The performance of the classifier with different features is shown in Table 4. Similar features as those we use, have been found to be adequate for activity and transportation modality classification [21, 22].

We found the results to be promising with just two features energy and standard deviation, hence the classifier trained with these features was implemented in the tag. Table 6 shows the performance of the classifier evaluated by real life experiments on raw unprocessed data. The performance of the classifier differs drastically in real life compared to WEKA. This is because we train and test using the same data set in WEKA. Since real life situations are very dynamic the type of movement the tag will encounter might differ from the trained data set, degrading the performance of the classifier. WEKA is still used as a guidance for building the classifier, keeping in mind that performance of



(a) Hermes Mote (b) Tv (c) Washing Machine

(d) Refrigerator (e) Hand (f) Vehicle

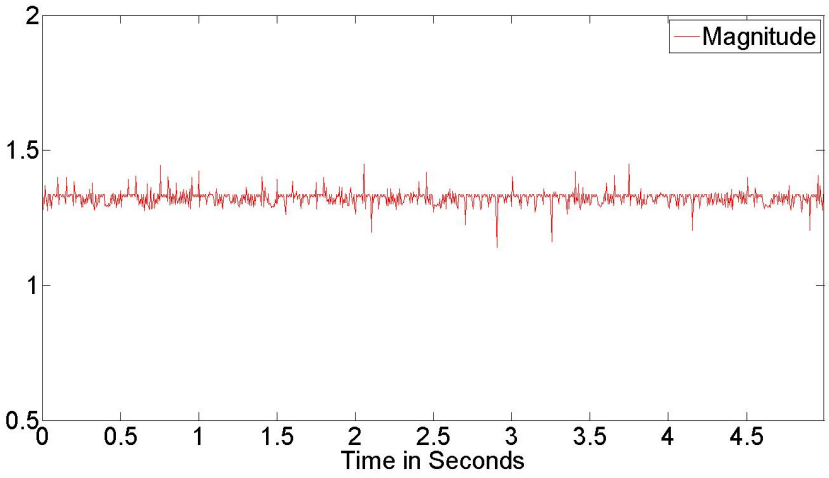
Figure 13: Prototype Hermes tag. (a) The tag attached to a television (b) Tag attached to a washer (c) Tag attached to a refrigerator (d) Tag attached to a persons hand (e) Tag attached to a vehicle

the classifier varies greatly in real life.

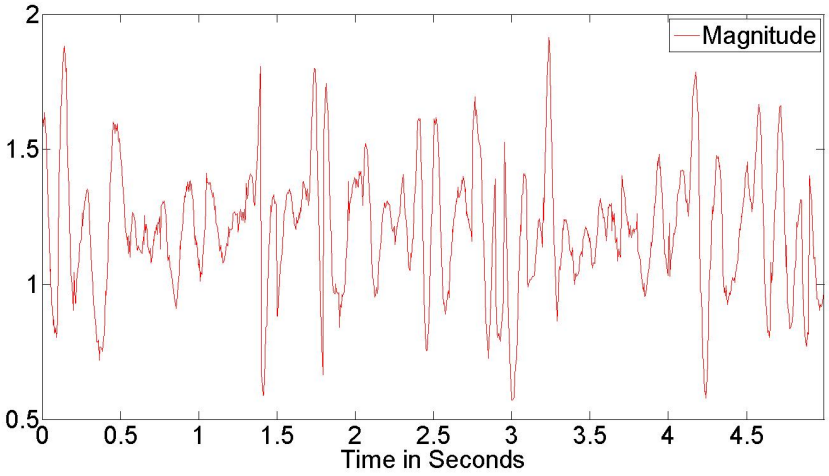
In Table 6, we find that the number of misclassifications of slow being classified as fast is 71.14% . This leads to an increase in false positives which causes the tag to send messages at higher frequency and eventually the tag dies quicker due to depletion of energy. This situation can be improved by preprocessing as explained in Section 5.1.3.

5.1.3 Data Pre Processing for Asset Track

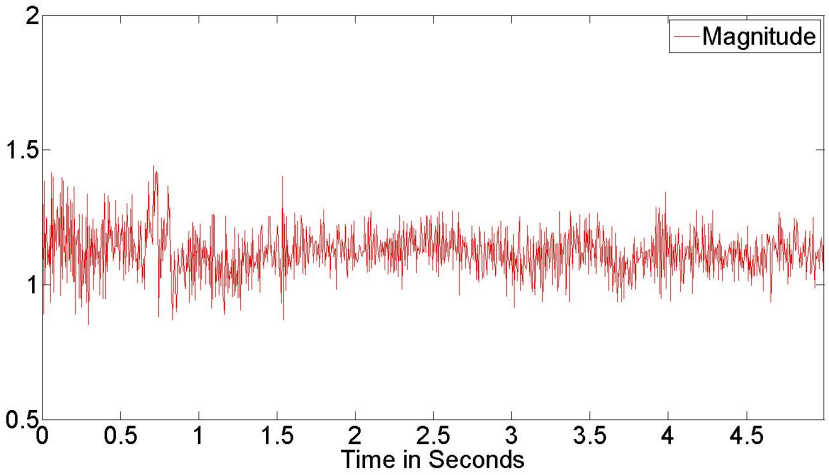
The collected raw data is usually noisy due to various external factors like wind, vibrations, jerks and sudden displacement. Removing these noises from the signals helps in building a reliable classifier. We eliminated these noises from the data by adding the preprocessing step. Preprocessing acts similar to a low pass filter by removing the outliers and jitters in the data. We tried different commonly used preprocessing methods in the literature out of which median over every 15 samples worked the best for us. The data set containing 210 data points for 1.05 seconds was then reduced to 14 data points. Figure 15(b) shows the smoothing of the signal after the median is applied retaining



(a) Magnitude of acceleration in units of 'g' of television data



(b) Magnitude of acceleration in units of 'g' of walking data



(c) Magnitude of acceleration in units of 'g' of car data

Figure 14: Magnitude of acceleration in units of 'g' for various activities

Table 4: The performance of various features on decision tree in WEKA.

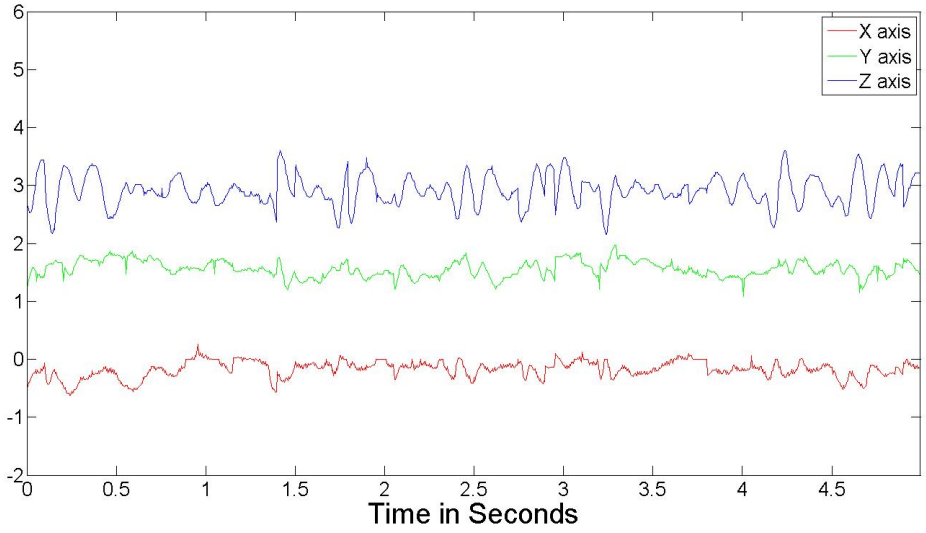
	2 Features	**3 Features**	**14 Features**
Correctly Classified Instances	95.74%	97.14%	98.99%
Incorrectly Classified Instances	4.26%	2.86%	1.31%

the nature of the signal. The size of the decision tree decreased by more than 60% after applying median to the data set. Figure 15(a) shows the stages of data preprocessing.

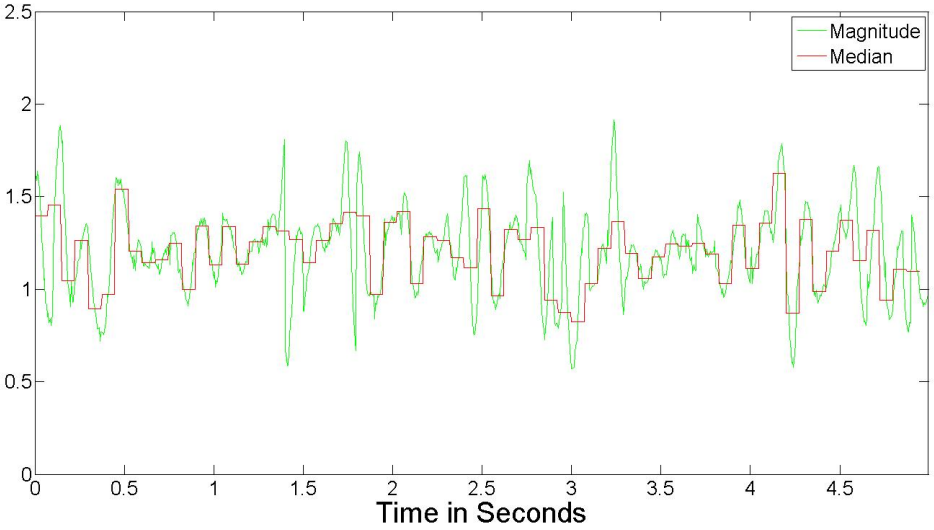
The performance of the classifier in WEKA with different features after applying median to the data set can be found in Table 5. We observe that the performance of the classifier has increased even with two features from 95.74% to 99.16% after preprocessing.

Figure 16(a) shows the 3D view of the scatter plot of feature values for the classes *slow*, *static* and *fast*. After the preprocessing step the best performing features seems to be energy and standard deviation and can be seen in Figure 16(b). In the feature list average absolute second difference did not lead to significant increase in accuracy as seen in figures 16(c) 16(d) where there is more overlap in the classes of interest causing misclassification.

We evaluated the performance of the classifier after preprocessing in real life scenarios and can be found in Table 7. We found that the performance of the classifier improved considerably. The classification error of slow being classified as fast decreased from 71.14% to 5.86%. We can still observe an error rate of 5.86% for slow being misclassified as fast. This would lead to sending messages at higher frequency which causes significant energy loss. We can also observe an misclassification error of 2.7% for fast movement being classified as slow. The performance of this classification is critical as it leads the tag to send messages at lesser frequency which causes an high chance to miss communication with the anchor node. We found that the classification error increased when the tags experienced transition from one state to another. As we mentioned in Section 5.1.2, each data point consists of 1.05 seconds of data. If the tag experienced a transition during that period then there is a high probability of misclassification. In order to improve the classification accuracy and avoid problems with transitions, we went for postprocessing.



(a) ADC x, y, and z axis of accelerometer



(b) Applying median over magnitude

Figure 15: Preprocessing stages of accelerometer signal

Table 5: The performance of various features on decision tree after median is applied in WEKA.

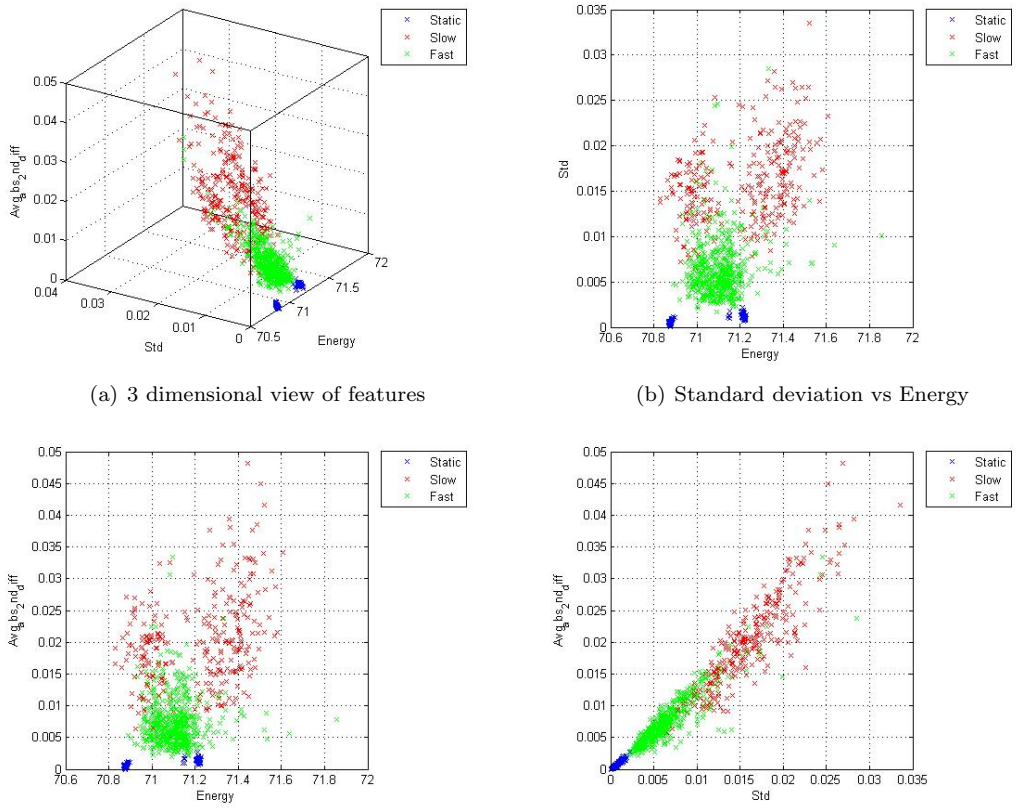
	2 Features	**3 Features**	**14 Features**
Correctly Classified Instances	99.16%	99.11%	99.36%
Incorrectly Classified Instances	0.832%	0.881%	0.6369%

5.1.4 Data Post Processing for Asset Track

A free moving object has the ability to change its state. Here by state we mean the static, slow or fast. Usually, during transitions between the states the object experiences 1 to 2 seconds of movement that is hard to classify into one of the states. The period of change in this state is usually very small and are called as transitions. We describe transitions as a state in which the object remains less than two seconds. Some examples of transitions may include walking to driving, driving to static, static to driving. By real life experiments we observed that the amount of classification error increased during transitions. This is because each data set sent for classification is of only 1.05 seconds. We handle this situation by post processing. To avoid false positives we used 5 consecutive data points to decide upon the class of movement that the tag belonged to. By this approach we validate the class of movement for a sufficient period of time before declaring the decision. We used this approach as a post processing step which improved the accuracy of the classifier by reducing the errors caused by transitions. The post processing step we use is a simple majority rule which looks for the 5 consecutive decisions which means decisions from continuous five seconds of data and outputs the class of interest that occurs the most. We tried two post processing schemes and found that majority of 5 seconds gave 97.3% accuracy compared to the alternate of five seconds approach

Table 6: Real life performance of AssetTrack classifier without pre and postprocessing.

	Fast	**Slow**	**Static**	**Total Instances**
Fast	4916(98.32%)	84(1.68%)	0(0%)	5000
Slow	3557(71.14%)	1443(28.86%)	0(0%)	5000
Static	0(0%)	0(0%)	5000(100%)	5000



(a) 3 dimensional view of features

(b) Standard deviation vs Energy

(c) Average absolute second difference vs Energy

(d) Average absolute second difference vs Standard deviation

Figure 16: Scatter plots of best performing features in AssetTrack (a) The 3 Dimensional view of a scatter plot with 3 distinct features (b) The 2 Dimensional view of Standard deviation Vs Energy (c) The 2 Dimensional view of Average absolute second difference Vs Energy (d) The 2 Dimensional view of Average absolute second difference Vs Standard deviation

which picks the alternate data set of 5 second data and outputs the majority in them which gave 94.25% of accuracy. In Figure 17 we can see that d1 to d5 are the 5 consecutive decisions of the classifier from 5 seconds of data. The majority of 5 decisions gives the final result Df. We take Df as the final output of *emove* in AssetTrack.

Table 8 shows the real life performance of the classifier after the post processing. We observed that the misclassification error of fast being classified as slow decreased from 2.7% to 0% and slow classified as fast from 5.86% to 2.9% after post processing. For every 1 second data we take approximately 1.2 seconds to compute the decision. So on an average the classifier takes 12 seconds to output the decision Df.

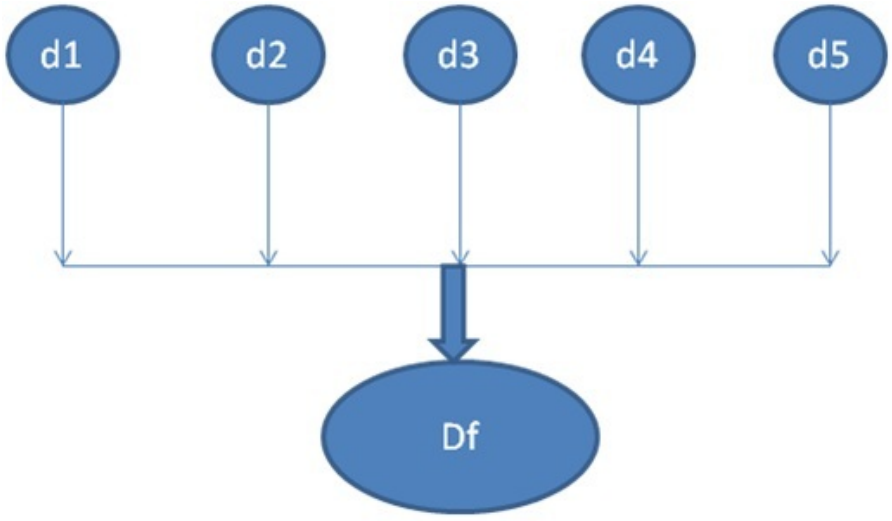


Figure 17: PostProcessing: Majority of five decisions.

5.2 AssetTrack System Development

Movement classification is based on two inertial sensors available on the tag-mote: Vibration dosimeter and 3-axis accelerometer. The vibration dosimeter detects when the objects is moving and wakes up the microcontroller. The operation of the vibration dosimeter is determined by hardware, and cannot be changed by the algorithm.

There is a vast study on classification of activity and movement using data from accelerometers. Most of such work focuses on healthcare applications, and estimation of energy expenditure [8,16,17, 25], classification of transportation modality [22, 23], movement classification for human-computer interfaces and smart environments [3, 21]. There is also much work on implementation of machine

Table 7: Real life performance of AssetTrack classifier with preprocessing.

	Fast	**Slow**	**Static**	**Total Instances**
Fast	4866(97.3%)	134(2.7%)	0(0%)	5000
Slow	293(5.86%)	4707(94.14%)	0(0%)	5000
Static	0(0%)	0(0%)	5000(100%)	5000

Table 8: Real life performance of AssetTrack classifier with pre and postprocessing.

	Fast	**Slow**	**Static**	**Total Instances**
Fast	1000(100%)	0(0%)	0(0%)	1000
Slow	29(2.9%)	971(97.1%)	0(0%)	1000
Static	0(0%)	0(0%)	1000(100%)	1000

learning algorithms on low power devices [4–6]. Following the literature, we use a hierarchical wakeup sequence, in which low power sensors are used to make an initial decision, and wake up other sensors, only if significant movement is suspected. This strategy reduces energy consumption. Specifically, the activation sequence is:

1. **Wake-up stage:** Vibration dosimeter is used to wake up the system from deep sleep mode. It is always active and wakes up the microcontroller after sustained vibrations.

2. **Movement detection:** An interrupt from the dosimeter wakes up the microcontroller and inturn the accelerometer. Acceleration measurements are used to decide if vibration dosimeter interrupt was caused by actual movement, or a short vibration produced, for example, by typing on a keyboard, or closing a microwave door.

3. **Movement classification:** Accelerometer measurements are used to classify movement as static, slow or fast as discussed in section 5.1 and perform the corresponding activity.

Notice that in each stage in this sequence the power consumption is higher than in the previous stages. The microprocessor, accelerometer and ADC together draws 1.62mA hence they should be activated only when a significant movement occurs. Figure 18 shows the block diagram of the classification system using vibra-tab and accelerometer in conjunction.

5.2.1 Wake-up Stage

The basic detection problem is to distinguish an object at rest from an object in (prolonged) motion while drawing less than 2 μA. We use a *vibration dosimeter*, shown in Figure 19, to perform this function. The sensor is an omni-directional vibration switch that is nominally closed at rest but chatters open and closed in response to movement [24]. The switch is connected to ground on one terminal and in series with a pullup resistor to power. The 2.49 MΩ pullup resistor sets the quiescent

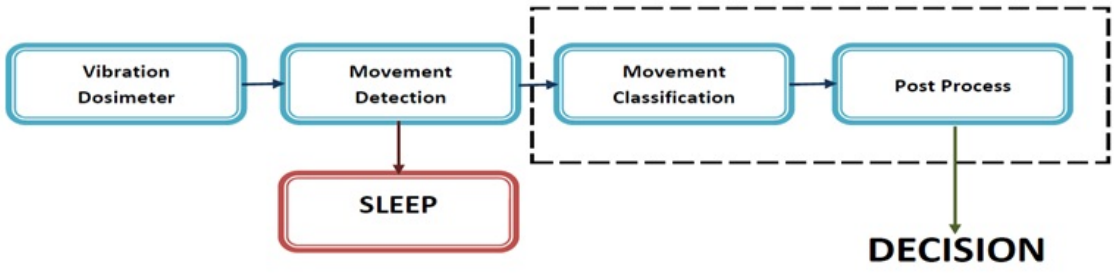


Figure 18: Block Diagram of Movement Classification

current draw of the circuit. At rest, the circuit draws 1.2 μA at 3 V. A capacitor AC-couples the output of the sensor, a first diode steers negative voltage transients to ground and a second diode steers positive transients to a capacitor that integrates these signals. A resistor in parallel with the integration capacitor slowly discharges the capacitor so that in the absence of motion, the capacitor voltage goes to zero.

Figure 20 shows the motion detector circuit in operation. It shows Tri-axial acceleration samples taken at 200 Hz are shown with their bias removed and amplitude scaled. The output of the motion detection wake-up circuit can be seen as a pulse that alternates between zero and one as the sensor transitions from rest to motion. At time t =0.5 s, a tag is picked up and moved, at time t =1.33 s, the motion detector circuit wake-up triggers, waking up the sleeping microcontroller using an interrupt line. At time t =3.09 s, the tag stops moving and time t =4.3 s, the motion detector output indicates movement has stopped. This process repeats for a second longer and more significant motion starting at time t =7.5 s.

Observations: According the specification [24] the vibratab switch is not guaranteed to be closed even when the sensing mechanism is under complete rest. That's the main reason for looking at the edge transitions like high-to-low and low-to-high rather than an open or closed state of the switch in our design. The Vibratabs can stay in a high state even if the objects stop moving. So edge transitions is observed for generating an interrupt. In our case we generate an interrupt every time there is an edge transition from low-to-high or high-to-low.

Vibratabs and its processing circuits work on nano power budgets as low as 0.25 μA. There are two vibratab which are added in hermes with different capacitor values such as 0.1 μA and 0.01μA. The one with capacitor value 0.1μA is the slow wake vibratab (V-SW), takes few seconds of motion

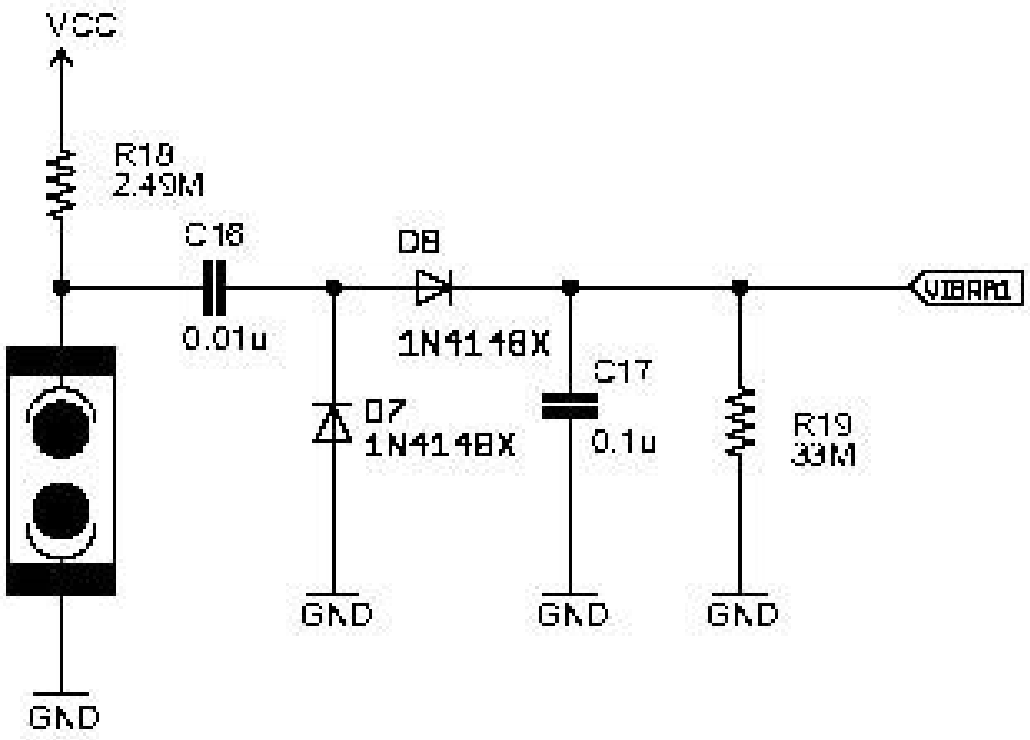


Figure 19: Motion detection circuit [10]. A dosimeter integrates the output of a vibration switch and trips after a brief period of continuous motion.

to generate an interrupt. The one with the capacitor value 0.01μA is the fast wake vibratab (V-FW), takes small and short motions to generate an interrupt. The first stage of the algorithm using the vibration dosimeter can be found below.

Algorithm 5.1 Moving Vs Static

This is algorithm executed on interrupt from vibration dosimeter

1: V-FW is enabled.
2: On V-FW interrupt
3: Disable V-FW.
4: movement ← Execute movement detection.
5: **if** movement=true **then**
6: Execute movement classification.
7: **else**
8: Go back to sleep. Enable V-FW.

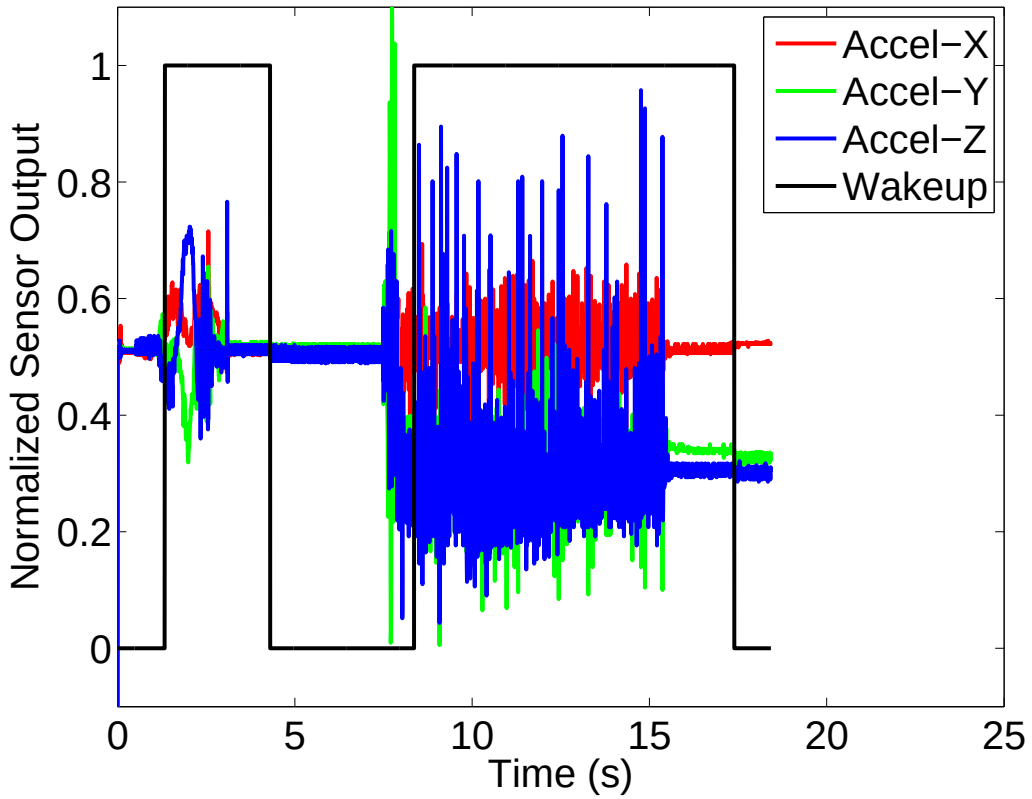
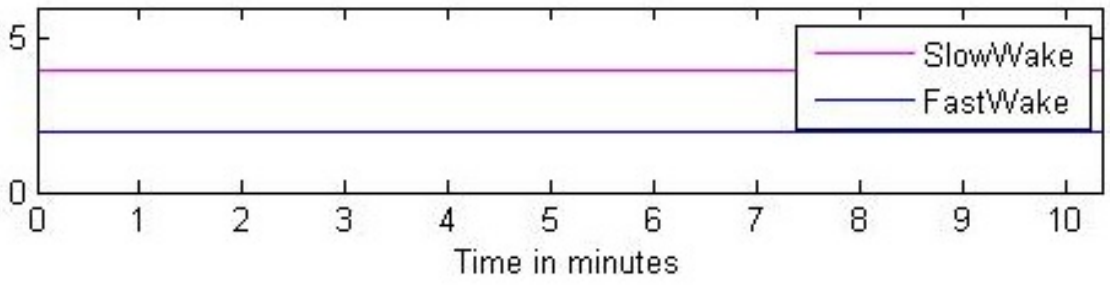


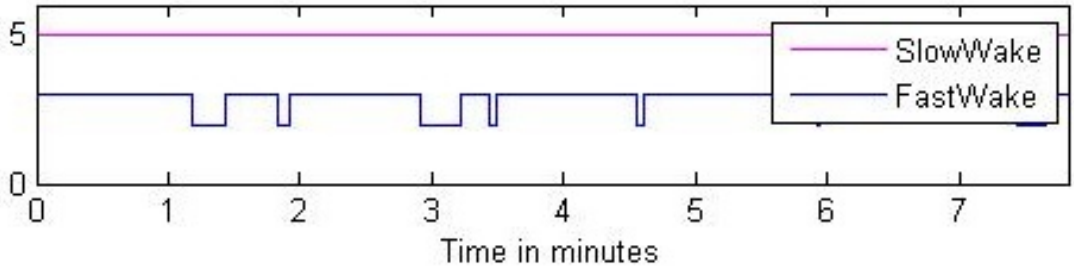
Figure 20: A example of the motion detection circuit in operation. Acceleration bias is removed and the readings are scaled. Figure courtesy Dr.Prabal Dutta, University Of Michigan.

Nature of vibratab signals

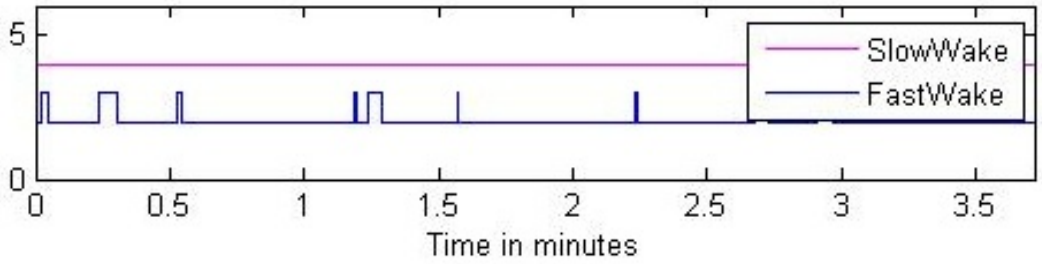
The tag should remain in the low power mode until it experiences an interrupt from the vibration dosimeter. The tag should not be sensitive to everyday vibrations of the objects it is attached to like televisions or washing machines or any natural vibrations like wind and earthquakes. Data was collected by sampling the ADC of the vibration dosimeter. This is mainly to understand the behavior of the vibration dosimeter under such conditions. We observe in Figure 21(a) there was no interrupt generated when the tag was deployed on a television. In Figure 21(c) and 21(b) we notice that the interrupt was generated by the vibratab in scenarios like walking and car respectively. We observe that the vibration dosimeter generates an interrupt only when there is a significant movement for a consistent period of time. It is observed from various real life observations that the average time to interrupt by the vibration dosimeter for any significant movement is 9 seconds and the maximum time observed was 18 seconds.



(a) Vibratab signal when placed on tv



(b) Vibratab signal when placed in car and driving on interstate



(c) Vibratab signal while walking

Figure 21: Nature of vibratab signals for various activities

5.2.2 Movement detection and classification

We now describe the details of the classification algorithm. The tag-mote is in *deep sleep* state until an interrupt is generated by the vibration dosimeter. The interrupt triggers the movement classification. The *movement classification phase* consists of a decision tree classifier, which determines if the tag-mote is in static, slow or fast based on features extracted from the medians. The classifier development is mentioned in Section 5.1. This classifier produces one decision for every second of accelerometer data. The classification algorithm is explained in 5.3. In post-processing phase, the decisions of 5 seconds are fed into a majority rule, which produces the final decision. The Post-processing algorithm can be found in 5.4. Post-processing helps reduce the false alarms produced by transitions, for example, when the tag-mote is initially static, and then is taken by a person and moved.

But executing the movement classification after an interrupt from vibration dosimeter seems to be less efficient in few scenarios. For instance (1) a short jerk would produce a interrupt from vibration dosimeter, but then, when the mote is awake, the movement would be gone. If we then run the classifier, the cost will be very high, because we have to sample for 5 seconds. Instead if we could verify the movement is consistent and is not caused by vibrations and jerks we could save energy spent in the classification phase. Hence movement detector was added between vibratab and movement classification phase. An interrupt from the vibratab the triggers the movement detection. The *movement detection* consists of a simple threshold rule: if the variance of the computed medians is below a threshold, it is assumed that the movement was caused by a short jerk or a vibration and the tag-mote returns to *deep sleep* state. Figure 22 shows the variance of static, slow and fast activity. It is clear from the graphs that variance of the accelerometer signal is higher during movement than when the mote is static (here static includes even the vibrations produced by television or stereo and not just complete lack of movement). The algorithm for movement detection can be found in 5.2. Movement detection increases the accuracy of classification and reduces the energy consumption by not triggering the movement classification stage when the object is really not moving.

Figure 23 shows the flowchart of the entire movement detection and classification of **emove**. It shows that after an interrupt is triggered from the vibratab the accelerometer is sampled and the ADC values are pre-processed. The processed data is then sent to movement detector. If the movement detector experiences a significant movement then it executes the movement classification. The movement classifier collects consecutive data for 5 secs and the decision is based on the majority rule.

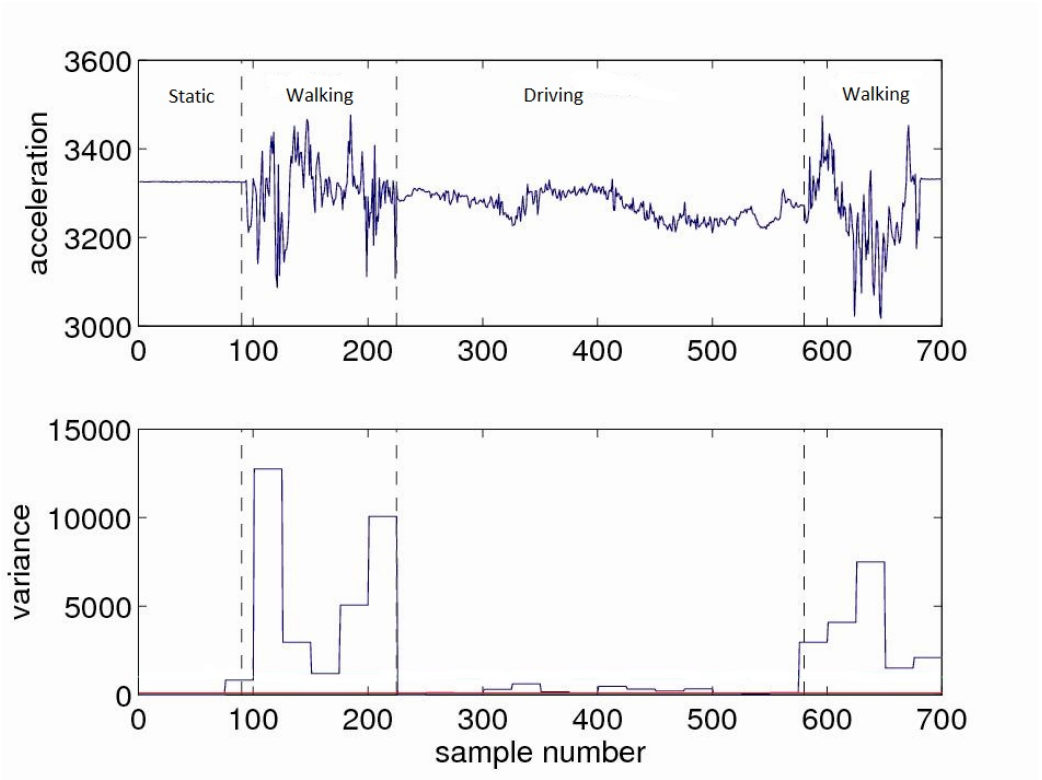


Figure 22: Usage of variance as movement detector

Algorithm 5.2 Movement Detection

Algorithm for movement detection using 3-axis accelerometer
Input: ADC of x, y and z axis of accelerometer **Output:** *Decision*

```
On interrupt from vibration dosimeter
rawdata ← Sample one second of ADC x, y, z axis of accelerometer at 200Hz
processedData ← preprocessor(rawdata)
variance ← computeFeatures(processedData)
Decision ← decisionTree(variance)
if Decision = TRUE then
    Execute Movement Classification
else Decision = FALSE
    goBacktoSleep
```

Algorithm 5.3 Movement Classification

Algorithm for activity classification using 3-axis accelerometer
Output: *Decision*

```
for i = 0; i ≤ 5; i++ do
    rawdata ← Sample one second of ADC x, y, z axis of accelerometer at 200Hz
    processedData ← preprocessor(rawdata)
    standarddeviation, energy ← computeFeatures(processedData)
    ListofDecisions[i] ← decisionTree(standarddeviation, energy)
Decision = CalculateMajority(ListofDecisions)
if Decision = static then
    return static
else if Decision = slow then
    return slow
else Decision = fast
    return fast
```

Algorithm 5.4 Post Processing: Calculate Majority

Finding the maximum occurrences in consecutive 5 decisions
Input: *ListofDecisions* **Output:** *static*, *slow*, *fast*

```
for i = 0; i ≤ length(listOfDecisions); i++ do
    if listOfDecisions[i] = STATIC then
        static++
    else if listOfDecisions[i] = SLOW then
        slow++
    else listOfDecisions[i] = FAST
        fast++
if static ≥ slow and static ≥ fast then
    return static
else if slow ≥ static and slow ≥ fast then
    return slow
else if fast ≥ static and fast ≥ slow then
    return fast
```

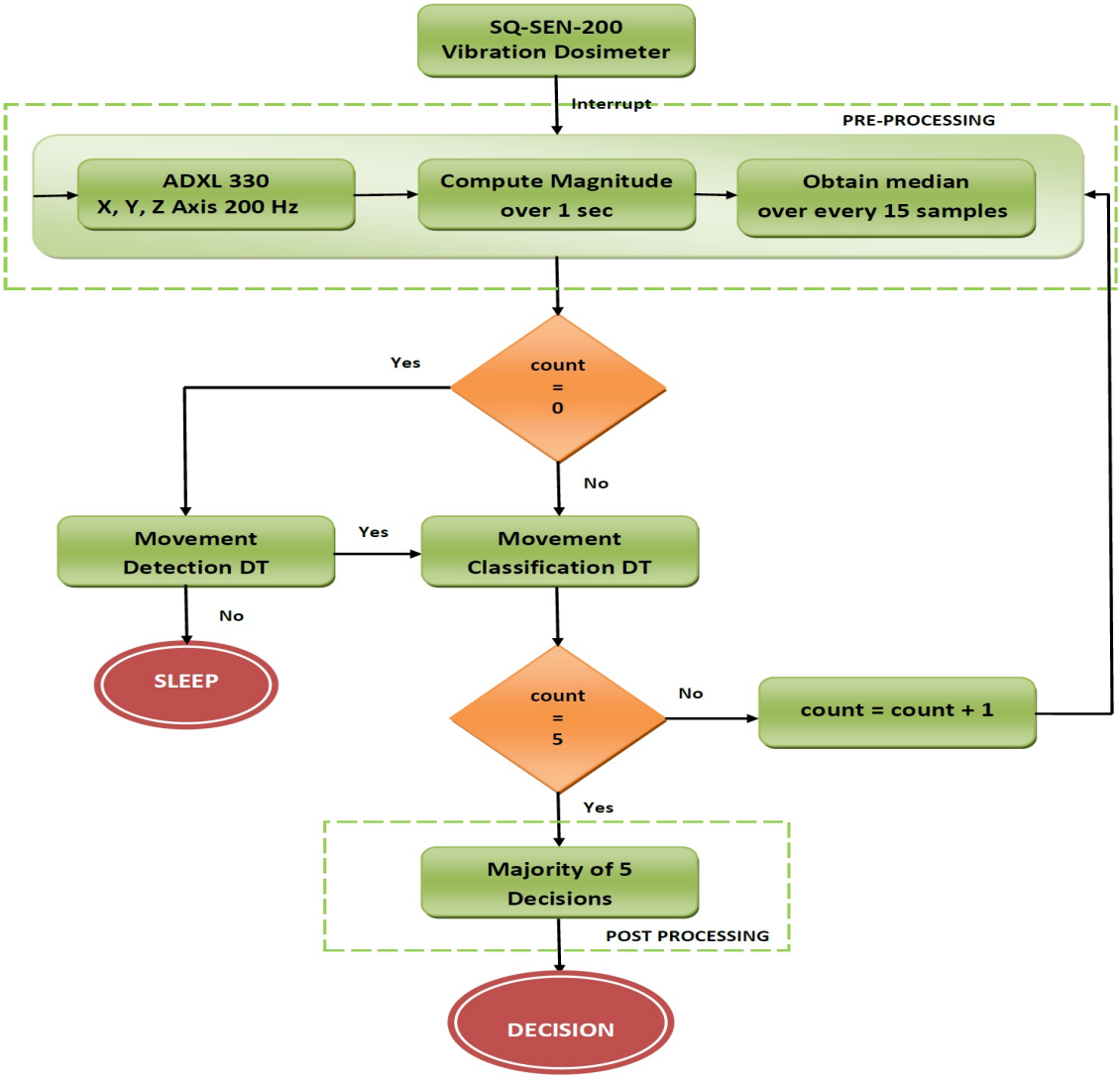


Figure 23: Flow chart of emove

5.3 Evaluation of AssetTrack

The goal of evaluation is to analyze the performance and accuracy of *emove* in AssetTrack by both simulation and real life experiments. Simulations was used during the development phase but experiments provided evaluation of the performance of the classifier in realistic conditions. As mentioned in Section 5.1, we classify movements into slow, static and fast. We implemented the algorithm described in Section 5.2.2 on a tag and evaluated it under various conditions by observing an LED and also by logging its result and then comparing it to the ground truth. The total number of samples (each sample represents 210 accelerometer readings or 1.05 seconds of data) used in the evaluation was $5,000$. To obtain classifier results from *static* situations in which the tag might wake up, we placed it on devices that may produce vibration such as televisions and speakers. To obtain *slow* movement data the tag was attached to a person, from whom activity like walking, running was logged. To obtain *fast* movement the tag was placed in a moving vehicle without any stops. The classifier was running on the tag, and its output was logged.

Classifier Evaluation: We compared the performance of the movement classification stage before preprocessing and after preprocessing and postprocessing stage in real life in three different classifiers. The first classifier was trained with raw acceleration data and did not include any median filtering or processing. The second classifier was trained with data preprocessed by applying the median over 15 samples. The third classifier was trained with preprocessed data and then postprocessing (i.e., majority rule) was applied to the classifier decisions. The results are presented in Table 9 in which each row corresponds to the ground truth, i.e., the actual type of movement of the tag. Each column corresponds to the estimated movement type (the output of the classifier). We observe that by doing preprocessing before extracting the features greatly reduces the number of misclassifications. We were able to achieve 97.32% accuracy for fast, 94.14% accuracy for slow and 100% accuracy for static during preprocessing stage. After postprocessing we were able to achieve 100% accuracy for fast and static classification and 97.1% accuracy for slow classification. Figure 25 shows the performance of the classifier at different stages.

Detection Time: We found that the time to detect a particular movement largely depends on the time to interrupt of the vibration dosimeter. We conducted several experiments under the activities of interest and logged every time an interrupt was generated. We found that the average time to interrupt was approximately 9 seconds. The frequency of interrupt decreases for activities which include less movement like the vibration caused by washing machine. There were hardly any interrupts observed. The frequency increases if the tag is experiencing a significant movement like walking or driving. The maximum time we observed for an interrupt to occur is 22 seconds. The

Table 9: Confusion matrices for the decision tree of AssetTrack classifier at different stages: Using only raw measurements (left), computing median over 15 samples (center), and computing median, and majority rule over 5 consecutive decisions (right). Row labels represent actual values, columns represent classifier output.

	Raw Data			**Preprocessing**			**Pre&Postprocessing**		
	Fast	Slow	Static	Fast	Slow	Static	Fast	Slow	Static
Fast	4916 (98.3%)	84 (1.68%)	0 (0%)	4866 (97.3%)	134 (2.7%)	0 (0%)	1000 (100%)	0 (0%)	0 (0%)
Slow	3557 (71.14%)	1443 (28.86%)	0 (0%)	293 (5.86%)	4707 (94.14%)	0 (0%)	29 (2.9%)	971 (97.1%)	0 (0%)
Static	0 (0%)	0 (0%)	5000 (100%)	0 (0%)	0 (0%)	5000 (100%)	0 (0%)	0 (0%)	1000 (100%)

time taken by the entire emove system is based on the sum of time taken by the vibration dosimeter say 9 seconds, time taken by the movement detector which is 2.2 seconds and time taken by the movement classification stage which is 11 seconds. It takes a total time of 22.2 seconds on an average and can take up to 35.2 seconds depending upon the delay caused by the vibration dosimeter.

Energy Evaluation: The computation of current draw by *emove* in AssetTrack largely depends upon the understanding of MSP430 microprocessors. For example, the MSP430 has one active mode (issuing instructions) and five low-power modes. The low power modes range from LPM0, which disables only the CPU and main system clock, to LPM4, which disables the CPU, all clocks, and the oscillator, expecting to be woken by an external interrupt source [26]. The power draws of these low power modes can differ by a factor of 350 or more (75μA for LPM0 at 3V, 0.2μA for LPM4). Correctly choosing the right microcontroller low power state can greatly increase system lifetime. By using the vibration dosimeter the system can operate under LPM4. The interrupt from the dosimeter wakes up the CPU and accelerometers. The power draw by the circuit in low power mode is sum of 0.2μA for LPM4 and 2μA for vibration dosimeter. We were not able to bring the current draw of Hermes to LPM4. As MSP430 defaults to LPM3 the maximum current draw in this mode is 4.3μA we take this mode as the base for our calculation. We take into account the sum of current draw of the circuit in LPM3 which is 4.3μA and vibration dosimeter 2μA which constitutes a total of 6.3μA in low power mode. In future if we can bring the circuit to LPM4 then the average current draw would be 2.25μA instead of 6.3μA. The current draw in movement detection and classification stage is 1.62mA which is the combined draw from microprocessor, accelerometer and ADC. The time

Table 10: Percentage of time the components are active in different phases of *emove*. The first column shows the percentage of time the components are active in the Deep sleep mode (DSM). The second column shows the percentage of time the components are active is Movement detection (MD) and Movement classification (MC). The third columns shows the current draw of the components.

	DSM	**MD & MC**	**Current Draw(μA)**
Vibration Dosimter	100%	100%	2
Accelerometer	0%	45%	320
ADC	0%	45%	800
Microcontroller	0%	100%	500

of operation for movement detection is 2.2 seconds and of movement classification is 11 seconds and the percentage of time each component is awake can be found in Table 10 and the charge drawn by the circuit at these stages are 0.6162μAh and 3.076μAh respectively. The life time of the tag node is shown in the Figure 24. We know that the minimum current draw of the tag as discussed above is 6.3μA in low power mode, hence the life time of the tag remains close to 3.6 years if its never moved. If the tag experiences jerks or vibration on an average of 100 times a day, then the life time of the tag would be closer 2.5 years. If the tag experiences significant movements like walking or running on an average of 100 times a day, then the life time would be closer to 1 year. As shown in the Figure 24 we can see that by adding movement detector stage the life time of the tag is increased approximately by 3 times. We can also see that the life time of the tag is increased from 5 days to 3.6 years by adding the vibration dosimeter as the first stage of detection. If the vibration dosimeter was not added then the tag could survive for only 5 days as the microcontroller, ADC and accelerometer would be on continuously. The life time of the tag was assessed based on the charge of the CR2032 battery which is 200 mAh. In Section 3 we mentioned that the life time of the tag can be closer to 10 years if the current draw of the circuit is less than 2μA. This can be achieved if the circuit operates on LPM4. We were able to bring down the minimum power draw of the circuit to only 6.3μA which we would like to address this issues as future improvement.

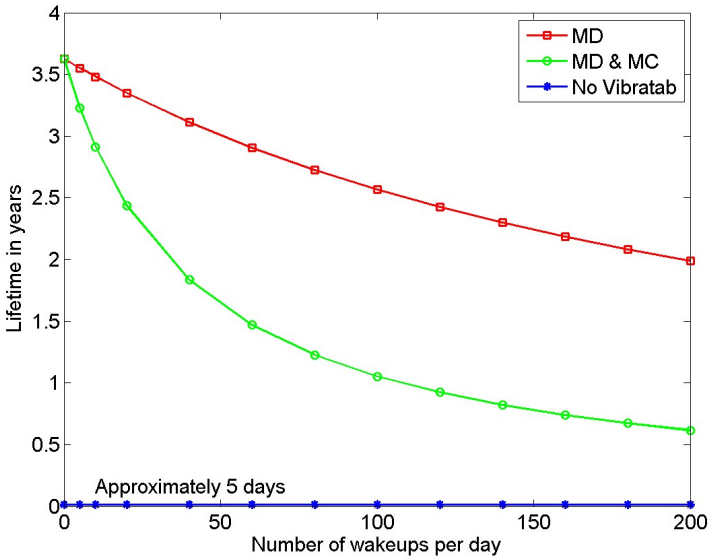


Figure 24: Life time of a tag at different scenarios.

6 AutoWitness

We implemented *emove* in another application called the AutoWitness for theft detection. AutoWitness is a system which is able to autonomously classify theft and track the stolen objects as the burglar moves through the city. The goal of the AutoWitness system is to act as a "rat" and eventually lead to the arrest of the burglar. AutoWitness takes into consideration a strong adversarial model and addresses both the issues arising from loss of radio signals or GPS connectivity. The system consists of two major components.

1. A battery powered tag node which can be embedded inside potentially lucrative objects in an incident of burglary.

2. An AutoWitness server for performing complex computations limited by the tag node's hardware capability.

Since most burglaries happen in the absence of the owners, AutoWitness tagged objects that are likely to remain in house are expensive. Such objects include safes, karoke systems, stereos, grand piano, desktops, antiques etc. The tag node is hidden in static objects which do not move very often.

Figure 26 shows the architecture of the AutoWitness system. On the left of this figure we show a tag node that is attached to a printer, to protect it against theft. The *emove* is implemented

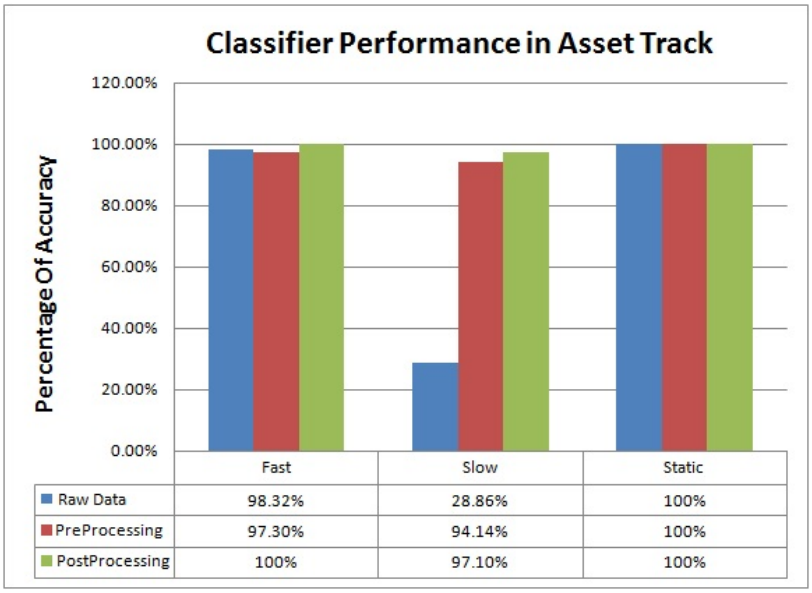


Figure 25: Performance of AssetTrack classifier at different stages of processing.

in the tag and is trained to identify theft. The system is designed to catch the organized burglars who use vehicles to transport the stolen objects from one place to another. AutoWitness system protects static objects which are medium to large in size. Such objects cannot be easily stolen by hand and needs a vehicle to transport them from one place to another. Hence, we associate vehicular movement to theft. The *emove* should be able to identify and classify theft from everyday movements and trigger tracking of the object only when it is being stolen. Once the tag detects a movement it executes the movement classifier to verify the type of movement. If the movement is identified as theft, it triggers the tracking algorithm, else it goes back to deep sleep. On detection of theft the burglar tracking module in the AutoWitness server receives the distance and turn estimates from the tag, which is used in tracing the route of the burglars vehicle. The server also informs the police of burglary and helps them catch the burglar with the live updates of the route from the server.

6.1 Classifier Development for AutoWitness

The AutoWitness classifier has only two classes of interest. They are: “vehicle” and “non-vehicle”. Figure 27 shows the 2 classes of interest and the activities associated with it. Object moving at a greater speed which cannot be achieved by a human activity is considered as vehicle. All the other

Figure 26: AutoWitness Architecture. Figure adapted courtesy Santanu Guha, University of Memphis.

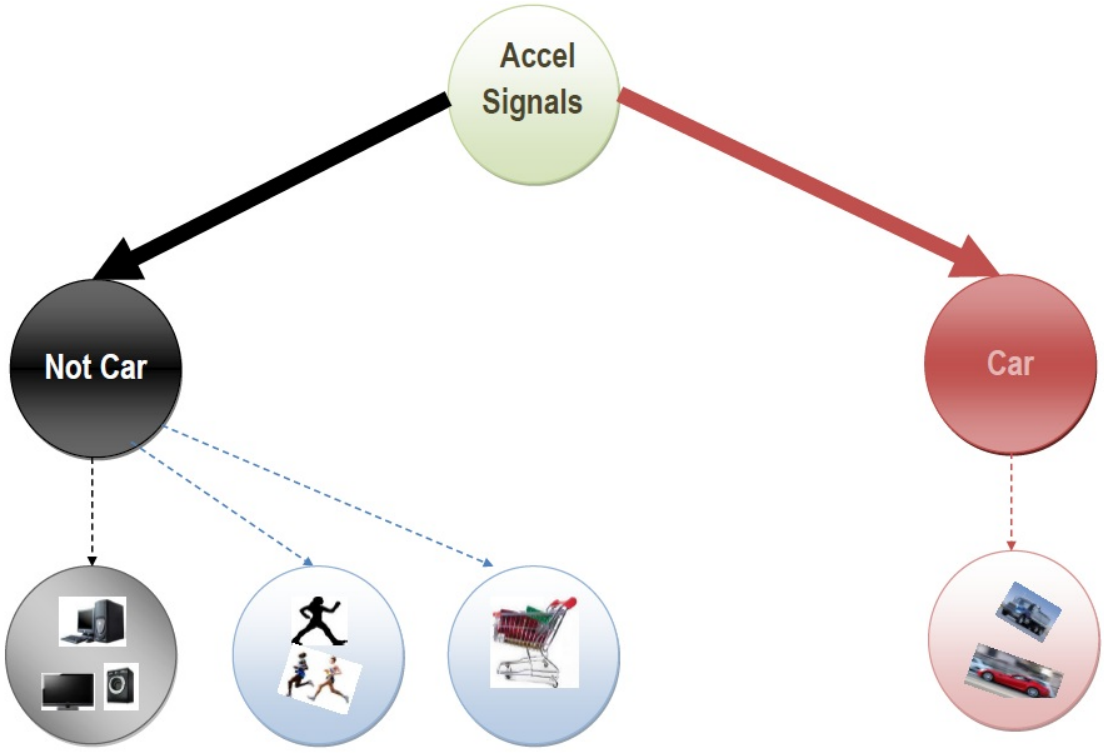


Figure 27: We can see the two classes of interest of AutoWitness and the activities that belong to the corresponding class.

movements like shaking, walking, running and static is considered a non-vehicle. One difficulty we found while developing the classifier is that there are activities which produce similar acceleration of a vehicle movement. For example activities like moving an object on a trolley or chair with wheels experience similar acceleration as in a vehicle. Identifying vehicular movement from such vast collection of other activities, is a challenge.

6.1.1 Data Collection

Data was collected extensively for vehicle and non vehicular activities. The data collected to represent non vehicular activities include activities such as walking, running, walking on stairs, jumping from different subjects, including male and female, as the tag should not identify such activities as theft. Figure 27 shows the classes of interest of AutoWitness and the activities corresponding to that class. The AutoWitness tag is usually deployed in appliances like washing machine, driers, television, stereo which vibrate while in operation. The tag should not wake up while such appliances

are in use. Therefore, data was collected from appliances when they were in use, and was added to the non vehicular class.

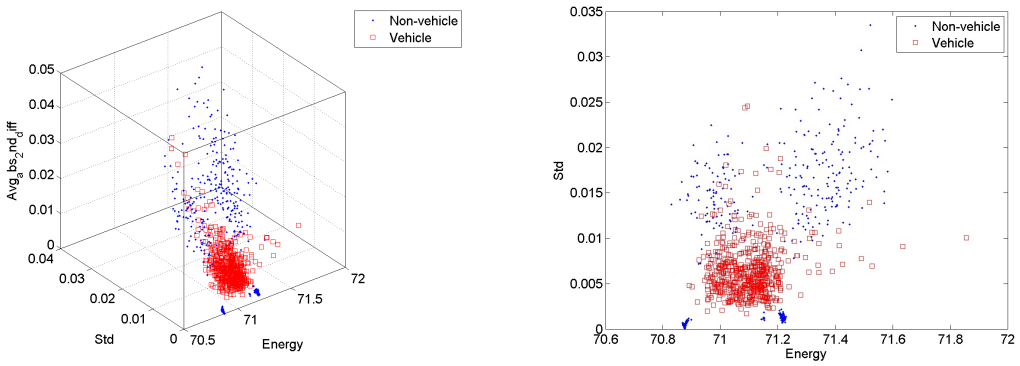
Since we are interested in vehicle signature than any other activities extensive scenarios of vehicular activity data were collected. They include driving on local roads, interstate, stop and go traffic, car static and engine on (to train the classifier with the low frequency vibration of the vehicle engine). The data was also collected while driving under different speeds ranging from 10mph to 70mph.

6.1.2 Feature Selection and Classifier Development

The collected data was preprocessed in the same way as in Section 5.1.3 and then used to train a decision tree in WEKA. We found that the best performing features were energy, standard deviation and average absolute second difference. The performance of the classifier does not vary much by removing average absolute second difference. Therefore we use only 2 features standard deviation and energy for training our decision tree, to keep the classifier as simple as possible. Table 11 shows that the performance of the classifier does not improve significantly by adding more features. Figure 28(a) shows the scatter plot of the 3 features in a three dimensional space. In Figure 28(b) we can see the scatter plot of standard deviation and energy. The dots closer to the x axis were generated by appliances while they were working or when they were static. The dots above the square are due to activities like walking, running, jumping. There are some dots overlapping the squares which are due to the activities overlapping the vehicular movement like asset moved on a trolley or asset moved on chair with wheels. The squares represent vehicular movement under various scenarios like interstate and local road driving. The decision tree generated in WEKA is used as the classifier for the movement classification phase of the algorithm.

6.1.3 Theft Detection System

The theft detection activation sequence is similar to the activation sequence of movement classification: (1) The low power vibration dosimeters wakes up the microcontroller when significant movement is detected (2) After waking up, the tag-mote activates the movement detector to verify that there is significant movement (3) In case of significant movement, it activates the movement classification, otherwise it goes back to sleep. The movement classifier samples the accelerometer, computes features and uses decision tree to detect if the tag is moving in a vehicle. If the decision is “vehicle” the tracking algorithm is triggered, else it goes back to sleep.



(a) 3D view of features

(b) 2D view of features

Figure 28: Scatter plots of best performing features in AutoWitness (a) Scatter plot of 3 feature values for vehicle and non-vehicle movement.(b) Scatter plot of 2 feature values for vehicle and non-vehicle movement.

6.2 Evaluation of AutoWitness

As mentioned in Section 6.1, we use vehicular movement as indicator of theft. We implemented the classifier described in Section 6.1 on a tag and evaluated it under various conditions by observing an LED and then comparing it to the ground truth similar to the AssetTrack evaluation. The total number of samples (each sample represents 210 accelerometer readings) used in the evaluation was $5,000$. We obtained vehicle movement data by placing the tag in a vehicle and driving without stops. To obtain non-vehicular data from walking activity a person walked with a tag in hand. To obtain data from static situations in which the tag might wake up, we placed it on devices that may produce vibration such as televisions and speakers. The classifier was running on the tag, and its output was logged.

Classifier Evaluation: We compared the performance of the movement classification stage in real life with three different types of classifier. The first classifier was trained with raw acceleration data and did not include any filtering or processing. The second classifier was trained with pre-processed data which is after applying the median over every 15 samples. The third classifier was trained with preprocessed data and then postprocessing was applied to the classifier decisions. The results are presented in Table 12. Each row in Table 12 corresponds to the ground truth, i.e., the actual transportation mode of the tag. Each column corresponds to the estimated transportation mode (the output of the classifier). Notice that the number of samples in the rightmost confusion matrix is less than the corresponding number in the leftmost matrix, because of the majority rule,

Table 11: Confusion matrices for decision tree of AutoWitness classifier, using 2, 3 and 14 features.

	2 Features		**3 Features**		**14 Features**	
	Vehicle	Non-Vehicle	Vehicle	Non-Vehicle	Vehicle	Non-Vehicle
Vehicle	1209 (98.16%)	9 (1.84%)	1884 (98.85%)	22 (1.15%)	1882 (98.74%)	24 (1.26%)
Non-Vehicle	9 (1.4%)	814 (98.6%)	43 (0.68%)	6240 (99.32%)	31 (0.5%)	6252 (99.5%)

Table 12: Confusion matrices for the decision tree of AutoWitness classifier at different stages: Using only raw measurements (left), computing median over 15 samples (center), and computing median, and majority rule over 5 consecutive decisions (right). Row labels represent actual values, columns represent classifier output.

	Raw Data		**Preprocessing**		**Pre&Postprocessing**	
	Vehicle	Non-Vehicle	Vehicle	Non-Vehicle	Vehicle	Non-Vehicle
Vehicle	4916 (98.32%)	84 (1.68%)	4866 (97.32%)	134 (2.68%)	1000 (100%)	0 (0%)
Non-Vehicle	3557 (35.57%)	6443 (64.43%)	293 (2.93%)	9707 (97.07%)	29 (1.45%)	1971 (98.55%)

which is the postprocessing step applied to 5 consecutive segments of one second. We observe that by doing preprocessing before extracting the features greatly reduces the number of misclassifications: only 2.93% of non-vehicle movement was classified as "vehicle," and only 2.68% of vehicle movement was misclassified. Applying the postprocessing eliminates false negatives entirely and limits false positives to $< 1.5\%$. Figure 29 shows the performance of the classifier at different stages. We were able to achieve 99.2% accuracy for the AutoWitness classifier.

Detection Time: The time to detect evaluation is also similar to the one explained in the Section 5.3. The average time to interrupt still remains 9 seconds and the time to detect an event on an average remains 22.2 seconds.

Energy Evaluation: The current draw in AutoWitness is similar to the AssetTrack and is described in Section 5.3 as we use the same stages as described in AssetTrack.

6.3 Stop Detector

A variant of *emove* is implemented as a "Stop-Detector" in the tracking phase of Auto-Witness. The detection of stops was necessary to find the distance between two consecutive stops in the tracking algorithm. The main requirements of a stop detector is to detect when a vehicle has stopped under any arbitrary situation, which mainly includes stops at traffic signals or intersections. The development of the stop detector is simpler than the other two applications. This is because the

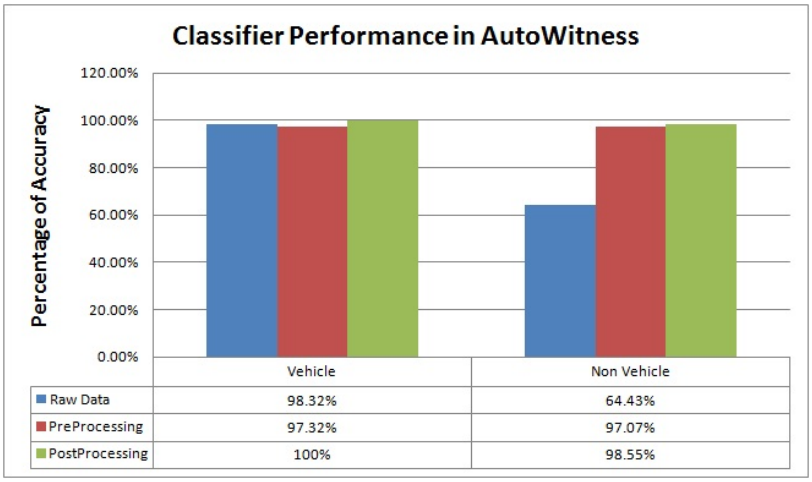


Figure 29: Performance of AutoWitness classifier at different stages of processing.

stop detector is activated in the active phase of the tracking, which uses accelerometer continuously, i.e, we don't need to worry about the wake-up sequence. The stop detector has continuous access to accelerometer readings. This eliminates the need to have the vibration dosimeter and movement detection phase of the *emove* system. We only need to train the classifier for the movement classification phase.

6.3.1 Classifier Development

There are only two classes of interest, "vehicle-moving" and "stop". Data was collected in different scenarios of vehicular movement like driving on local roads, driving on interstate at different speeds and in different vehicles. The stop data was collected in scenarios like tag placed in a stopped car, stopped car but engine "ON", stopped car on gear as these produced different low frequency vibrations which were critical in training the classifier. The tag experiences significant vibrations when the vehicle is not moving, if the engine is ON compared to the vehicle being static with engine turned off. The stop detector should detect when a moving vehicle makes a stop. Which means most of the time the engine will still be ON when it makes a stop. Using the data collected in the above mentioned scenarios helped in increasing the accuracy of classification.

6.3.2 Feature Selection

As before, we found that energy and standard deviation were the best features for the stop detection. The classifier was trained with these features in WEKA.

6.3.3 Stop Detection System

The accelerometer data is sampled continuously in the tracking phase. Every one second data is first pre-processed as explained in Section 5.1.3. After pre-processing, the features are computed from the data. The features are then supplied as input to the movement classifier. The decision tree of the stop detector for the movement classifier was generated by WEKA. The decisions are then post-processed giving the final result.

6.3.4 Evaluation

The stop detector was evaluated in real life by lot of experiments.

Classifier Evaluation: The stop detector was evaluated with three types of classifier. The performance of the stop detector only with pre-processing can be found in Table 13 and the performance of the stop detector after post processing is added, can be found in Table 14. We observe that the performance of stop detector has increased after adding post processing. We can see the performance of the classifier at different stages in Figure 30. We also conducted experiments in which we stopped 50 different times while driving and we found that the stop detector was able to detect 48 times giving us 96.95% accuracy in detection.

Detection Time: The time to detect a stop is of 11 seconds which is taken by the movement classification stage. As we do not use a vibration dosimeter and the movement detector the delay caused by these stages are eliminated.

Energy Evaluation: The current drawn by the stop detector is 1.62mA, hence the life time of the tag will be only 1 day if the stop detector of the vehicle is running continuously. The estimation of life time is only based on the current drawn by the stop detector and does not include the current drawn by the components in tracking phase. In that case the life time will further reduce.

Table 13: Real life performance of stop-detector with preprocessing

	Stop	**Vehicular Movement**	**Total Instances**
Stop	5000(100%)	0(0%)	5000
Vehicular Movement	793(15.86%)	4207(84.14%)	5000

7 Conclusions

In this thesis, we presented the design and evaluation of the *emove* system. We also showed its feasibility by evaluating the system in real life settings. Further, we have shown how *emove* can be applied in other systems such as AutoWiness as a theft detector and stop detector.

We developed *emove* to eliminate the need to communicate frequently to the infrastructure thereby reducing unwanted current draw and increasing the overall life time of the tag. *emove* detects and verifies if the movement is of interest and then performs the corresponding action according to the application its is used. The lifetime of the tag increases drastically when vibration dosimeter is added as a first level of detection before the accelerometer. As the vibration dosimeter draws current of only 2μA causing the entire circuit in low power mode to draw a total of 6.3μA which is relatively small. If the vibration dosimeter is removed then the current draw of the circuit would be 1.62mA which is approximately 250 times more. The dosimeter also acts an low cost motion sensor making the tag insensitive to natural vibrations and jerks. The current draw of a radio when sending a message is 17400μA which is relatively very high. Sending message periodically reduces the lifetime of the tag to days compared to 3.6 years with *emove*. By using vibration dosimeter there is an average delay of the detection of 9 seconds to a maximum of 22 seconds. So usage of the vibratab in conjunction with the accelerometer is purely left as design choice. If the application requires a

Table 14: Real life performance of stop-detector with preprocessing and postprocessing.

	Stop	**Vehicular Movement**	**Total Instances**
Stop	1000(100%)	0(0%)	1000
Vehicular Movement	61(6.1%)	939(93.9%)	1000

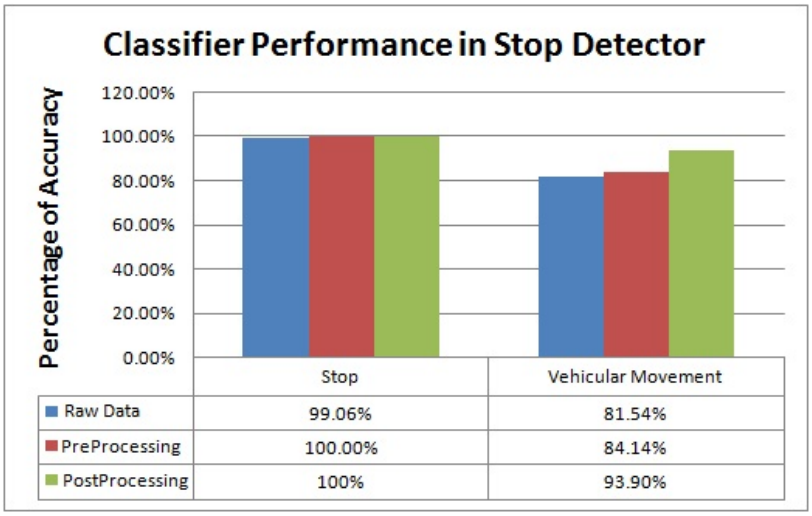


Figure 30: Performance of Stop Detector classifier at different stages of processing.

system to run in low energy mode to increase its lifetime and decrease the amount of current draw then usage of vibration dosimeter with accelerometer for movement classification works best. For some applications this delay might be costly and they might require the detection to be as quick as possible, in which case one needs to compromise on the lifetime of the tag and use acceleration alone for their movement classification.

We used a simple classifier because of the hardware constraints of the tag node and were able to achieve 98.02% accuracy. By using a more complex classifier we might achieve even higher accuracy. This is the current state of the *emove*, however many challenges still remain. The average current draw of the *emove* tag in low power mode is 6.3μA but can be brought down to 2.25μA which we would like to address as future improvement. We showed that adoption of *emove* for an asset tracking and theft detection application has increased the life time of the tag approximately from few months to more than a year.

References

[1] *LoJack Security System for Stolen Vehicle Recovery*, 1978 (accessed July 23, 2010). `http://www.lojack.com/car/Pages/car-works.aspx`.

[2] *Analog Devices Inc.,Santa Clara,CA, ADXL330 Datasheet*. 2005 (accessed August 16, 2010).

[3] L. Bao and S. Intille. Activity recognition from user-annotated acceleration data. *Pervasive Computing*, pages 1–17, 2004.

[4] A. Benbasat and J. Paradiso. Groggy wakeup-automated generation of power-efficient detection hierarchies for wearable sensors. In *4th International Workshop on Wearable and Implantable Body Sensor Networks (BSN 2007)*, pages 59–64. Springer.

[5] A. Benbasat and J. Paradiso. A framework for the automated generation of power-efficient classifiers for embedded sensor nodes. In *Proceedings of the 5th International Conference on Embedded Networked Sensor Systems*, page 232. ACM, 2007.

[6] C. Chen, J. Ma, and K. Yu. Designing energy-efficient wireless sensor networks with mobile sinks. In *Proceeding of the 4th ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2006.

[7] Chipcon. CC2420 Preliminary Datasheet, 2004 (accessed August 22, 2010). `http://inst.eecs.berkeley.edu/~cs150/Documents/CC2420.pdf`.

[8] S. Crouter, K. Clowers, and D. Bassett Jr. A novel method for using accelerometer data to predict energy expenditure. *Journal of Applied Physiology*, 100(4):1324, 2006.

[9] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley, 2 edition, November 2001.

[10] P. Dutta. *Irene Mote*, 2008 (accessed July 23, 2010). `http://www.eecs.berkeley.edu/~prabal/projects/epic/IRENE_B.pdf`.

[11] P. Dutta, J. Taneja, J. Jeong, X. Jiang, and D. Culler. A building block approach to sensornet systems. In *Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems (SenSys)*, 2008.

[12] Energizer. Energizer cr2032 product datasheet, 2009 (accessed October 19, 2010). `http://data.energizer.com/PDFs/cr2032.pdf`.

[13] S. Guha, K. Plarre, D. Lissner, S. Mitra, B. Krishna, P. Dutta, and S. Kumar. Autowitness: locating and tracking stolen property while tolerating gps and radio outages. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, SenSys '10, pages 29–42, New York, NY, USA, 2010. ACM.

[14] P. Ibach, V. Stantchev, F. Lederer, A. Weiß, T. Herbst, and T. Kunze. WLAN-based asset tracking for warehouse management. In *IADIS International Conference e-Commerce, Porto, Portugal*, 2005.

[15] N. Karmakar. Streamlining the Supply Chain With Electronic Commerce: A Key to Success in the Digital Economy. In *Proceedings of the IADIS International Conference WWW/Internet*, 2002.

[16] D. Keenan and F. Wilhelm. Classification of locomotor activity by acceleration measurement: validation in Parkinson disease. *Biomed. Sci. Instrum.*, 41:329–34, 2005.

[17] D. Klapper, J. Weaver, H. Fernandez, and L. Ohno-Machado. Classification of Movement States in Parkinsons Disease Using a Wearable Ambulatory Monitor. In *AMIA Annual Symposium*. American Medical Informatics Association, 2003.

[18] M. Liu, T. Yang, S. Alptekin, and K. Kato. Deploying the mobile-agent technology in warehouse management. *Intelligent Problem Solving. Methodologies and Approaches*, pages 651–659.

[19] M. Malinowski, M. Moskwa, M. Feldmeier, M. Laibowitz, and J. A. Paradiso. Cargonet: a low-cost micropower sensor node exploiting quasi-passive wakeup for adaptive asychronous monitoring of exceptional events. In *Proceedings of the 5th International Conference on Embedded Networked Sensor Systems (SenSys)*, pages 145–159, 2007.

[20] J. Polastre, R. Szewczyk, and D. Culler. Telos: Enabling Ultra-Low Power Wireless Research. In *Proceedings of the 4^{th} International Conference on Information Processing in Sensor Networks (IPSN/SPOTS)*, 2005.

[21] N. Ravi, N. Dandekar, P. Mysore, and M. Littman. Activity recognition from accelerometer data. In *Proceedings of the National Conference on Artificial Intelligence*, volume 20, 2005.

[22] S. Reddy, J. Burke, D. Estrin, M. Hansen, and M. Srivastava. Determining transportation mode on mobile phones. In *Proceedings of the 12th IEEE International Symposium on Wearable Computers*, 2008.

[23] S. Reddy, M. Mun, J. Burke, D. Estrin, M. Hansen, and M. Srivastava. Using mobile phones to determine transportation modes. *ACM Transactions on Sensor Networks (TOSN)*, 6(2):1–27, 2010.

[24] SignalQuest Precision Microsensors. SQ-SEN-200 Omnidirectional Tile and Vibration Sensor, 2009 (accessed June 21, 2010). `http://www.signalquest.com/sq-sen-200.htm`.

[25] A. SWARTZ, S. STRATH, D. BASSETT, W. O'BRIEN, G. KING, and B. AINSWORTH. Estimation of energy expenditure using CSA accelerometers at hip and wrist sites. *Medicine and science in sports and exercise*, 32(9):S450–S456, 2000.

[26] R. Szewczyk, P. Levis, M. Turon, L. Nachman, P. Buonadonna, and V. Handziski. *TEP:112*, 2007 (accessed August 22, 2010). `http://www.tinyos.net/tinyos-2.x/doc/html/tep112.html`.

[27] Texas Instruments. MSP430F11x2, MSP430F12x2 mixed signal microcontroller (Rev. D). Datasheet, 2004 (accessed August 22, 2010). `http://www.ti.com/lit/gpn/msp430f1232`.

[28] Weka Machine Learning Project. Weka. URL http://www.cs.waikato.ac.nz/~ml/weka, 2008 (accessed July 23, 2010).

[29] M. Youssef and A. Agrawala. The Horus location determination system. *Wireless Networks*, 14(3):357–374, 2008.

[30] WiseTrack Active RFID Asset Tracker, (accessed October 12, 2010). `http://www.wisetrack.com/rfid.htmll`.

[31] Gao RFID Asset Tracking, (accessed October 11, 2010). `http://www.gaorfidassettracking.com/About_GAO_RFID_Asset_Tracking`.

[32] TAGSYS RFID, (accessed October 12, 2010). `http://www.tagsysrfid.com`.